04C0
09.10.01

500.40572X00

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

#2

Applicant(s): N. SAGAWA.

Serial No.: 09 / 942,215

Filed: AUGUST 30, 2001

Title: TRANSACTION PROCESSING SYSTEM HAVING SERVICE LEVEL CONTROL CAPABILITIES.

LETTER CLAIMING RIGHT OF PRIORITY

Assistant Commissioner for
Patents
Washington, D.C. 20231

SEPTEMBER 20, 2001

Sir:

Under the provisions of 35 USC 119 and 37 CFR 1.55, the applicant(s) hereby claim(s) the right of priority based on:

Japanese Patent Application No. 2001 - 028231
Filed: FEBRUARY 5, 2001

A certified copy of said Japanese Patent Application is attached.

Respectfully submitted,

ANTONELLI, TERRY, STOUT & KRAUS, LLP

Carl I. Brundidge
Registration No. 29,621

CIB/rp
Attachment

日　本　国　特　許　庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office

出願年月日
Date of Application: ２００１年　２月　５日

出願番号
Application Number: 特願２００１－０２８２３１

出願人
Applicant(s): 株式会社日立製作所

２００１年　８月３１日

特許庁長官
Commissioner,
Japan Patent Office 及川耕造

出証番号　出証特２００１－３０７７４５３

【書類名】　特許願
【整理番号】　GM0011039
【提出日】　平成13年 2月 5日
【あて先】　特許庁長官殿
【国際特許分類】　G06F 15/00
【発明者】
　【住所又は居所】　東京都国分寺市東恋ヶ窪一丁目２８０番地　株式会社日立製作所　中央研究所内
　【氏名】　佐川　暢俊
【特許出願人】
　【識別番号】　000005108
　【氏名又は名称】　株式会社日立製作所
【代理人】
　【識別番号】　100075513
　【弁理士】
　【氏名又は名称】　後藤　政喜
【選任した代理人】
　【識別番号】　100084537
　【弁理士】
　【氏名又は名称】　松田　嘉夫
【選任した代理人】
　【識別番号】　100114236
　【弁理士】
　【氏名又は名称】　藤井　正弘
【手数料の表示】
　【予納台帳番号】　019839
　【納付金額】　21,000円
【提出物件の目録】
　【物件名】　明細書　1

【物件名】　図面　1

【物件名】　要約書　1

【プルーフの要否】　要

【書類名】　明細書

【発明の名称】　サービスレベル制御機構を有するトランザクション処理システム及びそのためのプログラム

【特許請求の範囲】

【請求項１】

１つまたは複数のサービスを提供し、該提供する各サービスには１つまたは複数のクライアントが接続可能なトランザクション処理システムであって、

前記提供するサービスに対応して定義された優先度情報を保持する優先度情報保持手段と、

前記優先度情報保持手段に保持された優先度情報を参照して前記クライアントからトランザクション処理システムに投入された処理要求の実行優先度を決定する実行優先度決定手段と、を有し、

該決定された実行優先度にしたがって処理を行うことを特徴とするトランザクション処理システム。

【請求項２】

前記実行優先度決定手段は、提供するサービスに対応して定義された優先度情報に高い優先度が指定されたサービスに対する処理要求に対して、高い実行優先度を与えること特徴とする請求項１に記載のトランザクション処理システム。

【請求項３】

１つまたは複数のサービスを提供し、該提供する各サービスには１つまたは複数のクライアントが接続可能なトランザクション処理システムであって、

前記提供するサービスに対応して定義された優先度情報を保持する優先度情報保持手段と、

前記提供するサービスに対してクライアントから送付された処理要求を、前記提供するサービス毎に順序付けて格納するキューイング手段と、

格納した処理要求の滞留情報を該キューイング手段から取得する滞留情報取得手段と、

前記優先度情報保持手段に保持された優先度情報及び前記取得された滞留情報を参照して前記クライアントからトランザクション処理システムに投入された処

理要求の実行優先度を決定する実行優先度決定手段と、を有し、

該決定された実行優先度にしたがって処理を行うことを特徴とするトランザクション処理システム。

【請求項４】

前記滞留情報取得手段は、

前記キューイング手段中に滞留している処理要求の数と、

前記キューイング手段中に滞留している処理要求の到着時刻と、

を取得することを特徴とする請求項３に記載のトランザクション処理システム。

【請求項５】

前記実行優先度決定手段は、

前記提供するサービスに対応して定義された許容待ち時間と、前記処理要求の滞留情報取得手段により取得された処理要求の到着時刻を比較することにより優先度を決定することを特徴とする請求項４に記載のトランザクション処理システム。

【請求項６】

１つまたは複数のサービスを提供し、該提供する各サービスには１つまたは複数のクライアントが接続可能であり、該提供する各サービスは１つまたは複数の実行モジュールより構成されるトランザクション処理システムであって、

前記各サービスを構成する１つまたは複数の実行モジュールの識別子を格納する手段と、

前記実行モジュールを記憶する記憶手段と、

該各実行モジュールの更新を前記識別子に基づいて管理する更新管理手段と、を有し、

該更新管理手段により該実行モジュールが更新された場合には、前記サービスに対応するトランザクションの開始に先立って、更新された実行モジュールを記憶手段へ配置することを特徴とするトランザクション処理システム。

【請求項７】

前記更新管理手段は、前記各サービス毎に、１つまたは複数の実行モジュールの更新と、実行モジュールの更新の検出とを排他的に実行することを特徴とする

請求項６に記載のトランザクション処理システム。

【請求項８】

１つまたは複数のサービスを提供し、該提供する各サービスには１つまたは複数のクライアントが接続可能なトランザクション処理システムであって、

該クライアントから前記提供するサービスに対して送付された処理要求を提供するサービス毎に順序付けて格納するキューイング手段と、

該キューイング手段に格納した処理要求の滞留情報を取得する滞留情報取得手段と、

前記各サービスに対するトランザクションのスループットを検出するスループット検出手段と、

前記各サービスに対してトランザクションの処理プロセスを割り当てるプロセス割り当て手段と、を有し、

該プロセス割り当て手段は、取得した処理要求滞留情報と検出したトランザクションスループットとを参照し、前記各サービスに対してプロセスの割り当てを決定することを特徴とするトランザクション処理システム。

【請求項９】

前記プロセス割り当て手段は、

前記キューイング手段に格納された処理要求が増加する場合には、割り当てるプロセス数を増加する一方、キューイング手段に格納された処理要求が減少する場合には、割り当てるプロセス数を減少することを特徴とする請求項８に記載のトランザクション処理システム。

【請求項１０】

前記プロセス割り当て手段は、前記各サービスに応じて付与された優先度順にプロセスの割り当てを行うことを特徴とする請求項８に記載のトランザクション処理システム。

【請求項１１】

１つまたは複数のサービスを提供し、該提供する各サービスには１つまたは複数のクライアントが接続可能なトランザクション処理をコンピュータに実行させるプログラムであって、

前記提供するサービスに対応して定義された優先度情報を優先度情報データベースへ格納する手段と、

前記クライアントから提供するサービスに対して送付された処理要求を、前記提供するサービス毎に順序付けてキューへ格納する手段と、

該キューに格納した処理要求の滞留情報を取得する手段と、

前記優先度情報および前記滞留情報を参照してトランザクション処理に投入された前記クライアントからの処理要求の実行優先度を決定する手段と

該決定された実行優先度にしたがってトランザクションの処理を行う手段とをコンピュータに機能させることを特徴とするプログラム。

【請求項１２】

１つまたは複数のサービスを提供し、該提供する各サービスには１つまたは複数のクライアントが接続可能であり、該提供する各サービスは１つまたは複数の実行モジュールより構成されるトランザクション処理をコンピュータに実行させるプログラムであって、

前記各サービスを構成する１つまたは複数の実行モジュールの識別子に基づいて各実行モジュールの更新状況を判定する手段と、

前記実行モジュールの更新が実行された場合には、サービスに対応するトランザクションの開始に先立って、更新された実行モジュールを記憶手段へ配置する手段とをコンピュータに機能させることを特徴とするプログラム。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本発明は、トランザクション処理システムに関し、特に、複数の顧客からのトランザクション処理を行うものに関する。

【０００２】

【従来の技術】

トランザクション処理システムは、主に金融取引や受発注処理など企業情報システムの基幹系において、多数の処理要求を効率よく、かつ整合性を保証しながら実行するためのシステムである。通常はクライアント・サーバ型のシステム構

成がとられ、クライアント（端末）が要求を発行し、サーバが必要に応じてデータベースにアクセスしながら業務処理本体を実行する。サーバにて実際の業務を実行する処理プログラムをサービスと呼ぶ。

【０００３】

また、トランザクション処理システムにおいてサービスを提供する側をサービスプロバイダと呼ぶ。例えば、銀行リテール業務では、ＡＴＭやテラーがクライアントであり、顧客口座データベースを有する基幹システムがサーバとなる。そして、銀行がサービスプロバイダであり、預金の引き出し、預け入れ業務などはサービスに相当する。

【０００４】

サーバ側で用いられるトランザクション処理ミドルウェアが、トランザクションモニタである。その役割は、主に以下の２点である。

【０００５】

（１）クライアントから送られる処理要求を受け付けてキューイングし、要求の優先度やサーバの混雑度を勘案しながら、適当なサーバプログラム（サービス）に順次制御を渡す。これにより、サーバ資源の有功活用を図る。

【０００６】

（２）処理中に生起するエラーや障害を検出し、処理が成功裡に完結した場合には結果の書き込み処理（コミット）を、完結しなかった場合にはキャンセル（ロールバック）や再起動（リラン）処理を行う。これにより、トランザクション処理の整合性を保証する。

【０００７】

ここで、トランザクションモニタの一般的な構成を図１８に示す。

【０００８】

トランザクションモニタ２０９はサーバ２１５上に位置し、処理要求を発行するクライアントプログラムはクライアント端末２２１、２２２上に位置する。

【０００９】

サーバ２１５は通常ＵＮＩＸサーバ、メインフレームコンピュータなどであり、クライアント端末はパーソナルコンピュータ、ＡＴＭ端末などである。

【００１０】

トランザクションモニタ２０９は、リクエストキュー２１０、２１１と、スケジューラ２０４、およびトランザクション実行部２０７を有する。

【００１１】

サービスに対する処理要求は一般にメッセージ（電文）の形でクライアント２２１、２２２からサーバ２１５に渡される。このため、トランザクションモニタ２０９は通信処理機能を有し、これを制御しながら処理メッセージを受け付ける。

【００１２】

受信されたメッセージは、トランザクションモニタ２０９内に処理要求（リクエスト）として格納される。通常複数のリクエストがトランザクションモニタ２０９内に滞留することになるので、リクエストは先入れ先出しのデータ構造であるキュー２１０、２１１を用いて入力された順番に格納される。格納されたリクエストは、トランザクション実行部２０７の資源（ＣＰＵ、メモリなど）が空き次第、格納した順にキュー２０１、２１１から取り出され、対応するサービスプログラム２０６によって処理される。

【００１３】

（スケジューリングと負荷分散）

リクエストをキューから取り出しサービスプログラムによる処理に移すことを、スケジューリングという。スケジューリングを効率よく行うことは、トランザクション処理システムの効率を上げるために重要である。

【００１４】

特にサービスを提供する資源（プロセッサ、サーバなど）が複数存在する場合、各資源に対しどのようにリクエストを割り振るかが処理効率を左右する。複数資源に対してリクエストを適切に割り当てトランザクションの処理効率を上げることを、負荷分散と呼ぶ。以下では、上記スケジューリングと負荷分散を含め、リクエストと資源との割り当て処理全般をスケジューリングと呼ぶことがある。

【００１５】

スケジューリングの方式としては、サービスを提供するプロセス数の増減によ

る負荷分散が知られている。図１９を用いてその概要を説明する。

【００１６】

図１９において、リクエストキュー３００には、リクエスト３０１～３０９が格納されており、これらはプロセス３１０～３１３によって順次処理される。ここでプロセスとはプログラムが計算機上で処理される単位であり、一つの仮想アドレス空間と、そこにローディングされたプログラム、データ、およびプログラム実行状態を示すＣＰＵレジスタの組よりなる。

【００１７】

図１９の例では、各プロセスには同一のトランザクション処理プログラム（サービスプログラム）がローディングされている。計算機のＣＰＵに空きがある間は、プロセス数を増やすことによって同時に実行されるサービス数を増やし、ＣＰＵの利用率を向上させることができる。

【００１８】

したがって、トランザクションに割り当てるプロセス数を増減することで、トランザクションの処理スループット（単位時間に処理されるリクエスト数）の調整が可能となる。

【００１９】

図１９（ａ）は、キュー３００にたまったリクエスト数が少ない状況を示す。トランザクションモニタは、リクエスト数に応じて、本サービスに少ないプロセス数（３１０、３１１）を割り当てる。

【００２０】

図１９（ｂ）は、多数のメッセージが到着し、キュー３００の待ち合わせリクエスト数が増加した状況を示す。トランザクションモニタはキューの状況を監視し、当該サービスに割り当てるプロセス数を増やす（３１０～３１３）。

【００２１】

図１９（ｃ）は、到着するメッセージが減少し、キューが短くなった状況である。トランザクションモニタは空いたプロセス３１３を当該サービスから除き、他のサービスまたは仕事に割り当てる。以上のように、キューの長さと割り当てるプロセス数を関連付けることによって、ＣＰＵリソースの範囲内でトランザク

ションの効率を向上させることができる。

【００２２】

また、トランザクションモニタの制御するサーバが複数存在する場合には、サーバ間の負荷分散として図２０に示す方式が用いられる。

【００２３】

今、サーバが３台（４２０～４２２）存在し、そのうち１台（サーバ４２０）のキュー４００が、サーバの処理能力や混雑度などの問題で他サーバのキュー４０１、４０２に比べて長くなったとする。クライアント側の処理プログラム４３１はこの状態を検出し、キューの短いサーバ４２１、４２２に対してメッセージを優先的に送付するように制御する。これにより、複数のサーバ間でのキューの長さが平均化され、総合的なスループットを向上させることができる。

【００２４】

（メッセージブローカ）

トランザクションモニタを応用してさらに高度な複合サービスを実現するシステムとして、メッセージブローカがある。

【００２５】

通常のトランザクション処理システムでは、一つのメッセージに対して一つのサービスが対応するのに対し、メッセージブローカでは、一つのメッセージを複数のサービス間で引き継ぎ、連鎖的に呼び出して処理を行う。あるメッセージに対し、どのサービスをどの順序で呼び出すかは、サービスフロー（業務フロー）としてトランザクションモニタ内に指定する。呼び出されるサービス自身は、トランザクションモニタと同じサーバにあっても良いし、他の独立なサーバにあっても良い。

【００２６】

メッセージブローカの構成を図２１に示す。

【００２７】

処理要求を発行するクライアントプログラム５０１、５０２はクライアント端末上に位置し、トランザクションモニタ５０９は、トランザクション処理用のサーバに位置する。

【００２８】

業務サービスを提供するサービスプログラムＡ５３０、サービスプログラムＢ５３１は、それぞれ別個のサーバ（あるいは同一のサーバでも良い）にローディングされる。端末および各サーバの間は、メッセージの通信路が確保されている。トランザクションモニタ５０９は、リクエストキュー５１０、５１１と、リクエストの処理順番を決定するスケジューラ５０４を有する。

【００２９】

通常のトランザクション処理システム（上記図１８）と比較して、メッセージブローカではトランザクション実行部（図１８の２０７）の機能が拡張され、サービスフロー実行部５０７を構成する。

【００３０】

サービスフロー実行部５０７においては、メッセージに応じて単なるサービスプログラムを起動、実行するのではなく、サービスプロバイダによって定義されたサービスフローの実行を管理する。

【００３１】

このようにメッセージブローカでは、トランザクションモニタ上でサービスフローの実行を可能とすることにより、複数のサービスプログラムを組み合わせてより複雑なサービスを構築することができる。

【００３２】

（稼働中のノード入れ替え）

メッセージブローカでは、業務の更新や追加によってサービスフローが頻繁に変更される可能性がある。サービスフローの変更のたびにシステム全体を停止させることは望ましくないので、システムを稼動させつつサービスフローのみを変更する仕組みが必要となる。

【００３３】

一つの方法は、サービスフローを実行するプロセス群を２つのグループ（現用グループと待機グループ）に分け、現用グループで変更前のフローを実行しつつ、待機グループ上に新しいサービスフローの再ローディングを行うことである。ローディングが終了した段階でメッセージのルーティングを現用グループから待

機グループへ切り替えて運用を続行する。

【００３４】

もう一つの方法は、サービスノード毎にルーティング可・不可の状態を設け、入れ替え対象ノードのルーティングを不可にし、ノードへのメッセージ入力を抑止した上でノードの再ローディングを行うことである。

【００３５】

上記のようなトランザクションまたはメッセージブローカの処理システムとしては、特開平９－６２６２４号公報（オンライントランザクション処理システム）、特開平６－２４３０７７号公報（分散型トランザクション処理システム）、特開平８－６３４３２号公報（優先順位を考慮したトランザクション一括処理システム）、特開平６－５２１２１号公報（バッチ処理、リアルタイム処理振り分け型トランザクション処理システム）、特開平７－７３１４３号公報（時間帯別優先順位制御のトランザクション処理システム）、特開平１０－４０１１７号公報（レスポンスを維持するタスク制御式のオンライントランザクション処理システム）等が知られている。

【００３６】

【発明が解決しようとする課題】

複数のサービスプロバイダ（または顧客）のシステムをアウトソーシング形態で受託し、計算機資源を集中管理することにより全体としての処理効率を向上させるデータセンタなどの業態が盛んになりつつある。

【００３７】

データセンタでは、委託元であるサービスプロバイダとの保証契約（サービスレベルアグリーメント：ＳＬＡ）により、利用した各計算機資源（トランザクション量やそれに伴うＣＰＵ時間、データ量など）やサービスの保証形態に応じて課金を行い、対価を請求する。課金を廉価に押さえるためには、より少ない計算機資源（投資）でより多くのトランザクションを実行する必要がある。

【００３８】

これに対し、上記従来のトランザクションモニタやメッセージブローカ等のトランザクション処理システムにおいては、単一のサービスプロバイダがクライア

ントにサービスを提供することを目的に作成されてきたため、一つのトランザクション処理システムを複数のサービスプロバイダが共用するような形態において、サービスプロバイダ間でのトランザクション資源（計算機資源）やスループットを調整するための配慮がなされていなかった。

【００３９】

このため、複数のクライアントからトランザクション処理要求があった場合には、計算機資源の適切な利用を図ることができず、スループットを保証するのが難しいという問題があった。

【００４０】

また、上記従来のメッセージブローカやトランザクションモニタでは、業務の更新や追加によるサービスフローの更新に際し、予備系プロセスを設けるか、あるいはルーティングを閉止するなどの処置が必要であり、計算機資源を有効に利用するための配慮が為されておらず、柔軟に運用することが難しいという問題があった。

【００４１】

そこで、本発明の目的は、上記ＳＬＡを勘案したトランザクションの優先度制御、および計算機資源の割り当て制御を可能とすることにより、複数のサービスプロバイダに対するサービスを提供するのに好適なトランザクション処理システムを実現することにある。

【００４２】

【課題を解決するための手段】

本発明のトランザクション処理システムは、サービスに対する優先度制御を実現するために、

トランザクション処理システムが提供するサービスに対応して定義された優先度情報を格納する優先度情報保持手段と、提供するサービスに対してクライアントから送付された処理要求を、該提供するサービス毎に順序付けて格納するキューイング手段と、該キューイング手段から該格納した処理要求の滞留情報を取得する滞留情報取得手段と、該クライアントから該トランザクション処理システムに投入された処理要求の優先度を、優先度情報および処理要求の滞留情報を参照

して、その実行順序を決定する実行優先度決定手段とを有する。

【００４３】

より好ましくは、前記キューイング手段は、サービスを提供する顧客または利用者毎に処理要求を格納するキューを複数備え、また、優先度情報保持手段は、処理（実行するサービス）の種類と顧客または利用者に応じて定義付けられた優先度情報を有するものである。

【００４４】

また、本発明のトランザクション処理システムは、上記サービスに対する計算機資源の割り当て制御を行うために、各サービスに対するトランザクションスループットを検出する手段と、該各サービスに対してトランザクション処理プロセスを割り当てる手段と、を有し、該プロセスを割り当てる手段は、前記各サービスに対して、該取得した処理要求滞留情報と該検出したトランザクションスループットとを参照してプロセスの割り当てを決定する。

【００４５】

また、本発明のトランザクション処理システムは、各サービスを構成する実行モジュールの更新に応じて記憶手段への配置（ローディング）を行うために、

各サービスを構成する１つまたは複数の実行モジュールの識別子を格納する手段と、該各実行モジュールの更新を前記識別子に基づいて管理する更新管理手段と、を有し、該更新管理手段により該実行モジュールの更新が実行された場合には、該各サービスに対応するトランザクションの開始に先立って、該更新された実行モジュールを記憶手段へ配置する。

【００４６】

【発明の効果】

以上のように、本発明のトランザクション処理システムまたはメッセージブローカでは、各サービスに対する優先度制御を実現するために、

トランザクション処理システムが提供するサービスに対応して定義された優先度情報と、クライアントから提供するサービスに対して送付された処理要求を該提供するサービス毎に順序付けて格納するキューイング手段から取得する処理要求の滞留情報とを勘案する。

【００４７】

このため、トランザクション処理システムが提供する顧客毎の各サービスについて、契約によって定義された条件を満たすようなトランザクションのスケジューリングが可能となり、より少ない計算機資源で、より多くのオンライントランザクションを、顧客との保証契約に応じたスループットを確保しながらリアルタイムで処理することが可能となり、複数の顧客の業務を集中して処理するデータセンタの信頼性と処理能力を向上させることができるのである。

【００４８】

また、本発明のトランザクション処理システムは、各サービスに対応するトランザクションのスループットを検出する手段と、該各サービスに対してトランザクション処理プロセスを割り当てる手段と、を有し、該プロセスを割り当てる手段は、各サービスに対して、取得した処理要求滞留情報と該検出したトランザクションスループットとを参照してプロセスの割り当ての決定を行う。

【００４９】

このため、トランザクション処理システムが提供する各サービスについて、契約によって定義された条件を満たすようなプロセスの割り当てが可能となって、顧客との保証契約に応じたスループットを確保しながらリアルタイムで処理することができ、複数の顧客の業務を集中して処理するデータセンタの信頼性と処理能力を向上させることができるのである。

【００５０】

さらに、本発明のトランザクション処理システムは、各サービスを構成する１つまたは複数の実行モジュールの識別子を格納する手段と、該各実行モジュールの更新を前記識別子に基づいて管理する更新管理手段と、を有し、該更新管理手段により該実行モジュールの更新が実行された場合には、該各サービスに対応するトランザクションの開始に先立って、該更新された実行モジュールを記憶手段へ配置する。

【００５１】

このため、実行モジュールが更新されているときには、各クライアントのトランザクション処理開始に先立って、更新された実行モジュールを記憶手段にロー

ドしてから実行するので、稼働状態を維持したまま業務の更新や追加を行うことが可能となって、柔軟性、可用性を高めることができ、また、実行モジュールの更新に際して、予備系プロセスを設けたりルーティングを閉止するなどの処置が不要となるため、計算機資源の有効利用を実現できる。

【００５２】

【発明の実施の形態】

以下、本発明の一実施形態を、添付図面に基づいて説明する。

【００５３】

１．ハードウェア構成

本発明の実施に好適な計算機システムのハードウェア構成例を図２に示す。システムは、１台または複数のクライアント端末７０１、７０２，１台または複数のトランザクション処理サーバ７０３、１台または複数のサービスプログラム実行用サーバ７０４、７０５より構成される。ただし、トランザクション処理サーバとサービスプログラム実行用サーバは同一の計算機であっても良い。

【００５４】

クライアント端末７０１、７０２はＡＴＭ（Ａｕｔｏｍａｔｉｃ　Ｔｅｌｌｅｒ　Ｍａｃｈｉｎｅ）端末やＭｉｃｒｏｓｏｆｔ　Ｗｉｎｄｏｗｓオペレーティングシステム、Ｌｉｎｕｘオペレーティングシステムなどの稼動するパーソナルコンピュータでも良い。

【００５５】

トランザクション処理サーバ７０３、サービスプログラム実行用サーバ７０４、７０５は、例えば、日立３５００シリーズのようなＵＮＩＸサーバ、日立フローラ（登録商標）シリーズのようなＷｉｎｄｏｗｓ　ＮＴサーバ（登録商標）、あるいは日立ＭＰシリーズのような大型汎用計算機で良い。クライアント、各サーバを結合する通信路７１０は、例えばＥｔｈｅｒｎｅｔのような汎用的なネットワークで良い。なお、トランザクション処理サーバ７０３、サービスプログラム実行用サーバ７０４、７０５には、図示しないメモリやハードディスク装置などの記憶手段が設置されている。

【００５６】

２．発明の概要

本発明に好適な実施形態の詳細を説明するに先立ち、本発明全体の概要を図１を参照しながら説明する。

【００５７】

クライアントプログラム１０１、１０２は、クライアント端末７０１、７０２上で稼動し、システムの末端利用者にインタフェースを提供するプログラムである。なお、クライアントは、トランザクション処理サーバ７０３と接続する顧客毎のクライアント端末７０１、７０２からなる。

【００５８】

ＡＴＭの制御プログラムや、パーソナルコンピュータ用のクライアントプログラムがこれに相当する。クライアントプログラムはｗｅｂブラウザであっても良い。クライアントプログラムは末端利用者の入力を受け付けて、メッセージを組み立て、トランザクションモニタ１２０に送信する。

【００５９】

トランザクションモニタ１２０は本発明の中心を構成する部分である。前記従来のトランザクションモニタとは異なり、本発明のトランザクションモニタは、複数のサービスプロバイダ（顧客、以下同様）に対するメッセージ（処理要求）を受け付ける。図１では、サービスプロバイダの数を２と仮定している。

【００６０】

ＳＬＡデータベース（優先度情報データベース）１１３は、各サービスプロバイダとのサービスレベル（優先度情報、許容待ち時間）に関する契約情報（ＳＬＡ）を格納する。例えば、「サービスプロバイダＡのトランザクションは、１０秒以内に処理されること」などの契約内容に基づいて、許容待ち時間が１０msec、優先度がＵ．Ｌなどの情報が本データベースに格納される。

【００６１】

メッセージディクショナリ１１４には、サービスプロバイダ毎のメッセージのフォーマットを定義する。例えば「プロバイダＡのメッセージの第１０バイトから第２０バイトまでの間は、顧客口座番号である」などの定義が格納される。

【００６２】

サービスフロー定義１１５には、サービスプロバイダ毎のサービスフローの定義が格納される。実行モジュールライブラリ１１６には、サービスフローの各サービスノードに対応した実行モジュール群が格納される。

【００６３】

プリプロセッサ１０３は、クライアントプログラム１０１、１０２から受け付けたメッセージを解釈し、該メッセージがどのサービスプロバイダに属するかを判定する。

【００６４】

リクエストキュー１１０、１１１は、該トランザクションモニタ１２０を利用するサービスプロバイダ毎（顧客毎）に設けられ、それぞれのサービスプロバイダに対して送られたリクエストを格納する。図１ではサービスプロバイダの数を２と仮定しているので、リクエストキューは１１０、１１１の２本が存在する。

【００６５】

キューイング状況検出部１１２は、リクエストキュー１１０、１１１を監視し、その状況（滞留リクエスト数やスループット）を取得する。

【００６６】

スケジューラ１０４は、キューイング状況検出部１１２から得られるキューイング状況と、ＳＬＡデータベース１１３に格納されたＳＬＡ契約とを勘案してスケジューリングの優先度を決定する。スケジューラ１０４はまた、サービスプロバイダ毎に割り当てるプロセス１０８、１０９の数を管理し、ＳＬＡ契約を満たすために適切な数のプロセス数を決定する。

【００６７】

スケジューラ１０４により取り上げられたメッセージは、ダイナミックローダ１０５に送られる。

【００６８】

ダイナミックローダ１０５は、サービスフロー定義１１５を参照し、現在のメッセージに対応するサービスフローを決定する。

【００６９】

実行モジュール管理部１０６は、実行モジュールライブラリ１１６を監視し、

変更が加えられた場合にこれを検出する。ダイナミックローダ１０５はこの検出結果を参照し、現在のメッセージに対応するサービスの実行に必要なサービスノードが当該プロセス中にすでにローディングされているかを判別し、ローディングされていない（あるいは古いモジュールがローディングされている場合には、新たなモジュールをローディングする。サービスフロー実行部１０７は、スケジューリングされたサービスフローを実行する。

【００７０】

以下に、本発明を構成する各要素の詳細な実施形態について説明する。

【００７１】

３．ＳＬＡデータベース

図３を参照して、ＳＬＡデータベース１１３の構成例を説明する。

【００７２】

ＳＬＡデータベース１１３は表の形でディスク上に格納され、トランザクションモニタ１２０を運用するデータセンタが、顧客（サービスプロバイダ）から受託した契約内容を蓄積する。

【００７３】

表の列はサービスプロバイダ名称８０１、サービス名称（処理の種類）８０２、クラス８０３、上限値８０４、優先度８０５を含む。

【００７４】

サービスプロバイダ名称８０１は、本トランザクションモニタの処理契約対象となっているサービスプロバイダを特定する名称である。本カラムは、一意の名称であれば任意の文字列で良い。

【００７５】

サービス名称８０２は、当該サービスプロバイダが本トランザクションモニタで提供するサービスの名称である。クラス８０３は、サービスプロバイダとの契約の種別を表す。「Ｂ．Ｅ．」はベストエフォートの略であり、リソースに空きがある場合にはトランザクションをスケジュールする契約を示す。リソースが込み合っていた場合、トランザクションは長い間待たされる可能性がある。「Ｕ．Ｌ．」はアッパーリミットの略であり、トランザクション待ち合わせの上限時間

を決める契約を示す。

【００７６】

上限値８０４は、Ｕ．Ｌ．契約における上限時間を示す。Ｂ．Ｅ．契約時にはこのカラムは意味を持たない。

【００７７】

優先度８０５は、Ｕ．Ｌ．契約されたサービスの間での優先順位を示す。リソースが込み合ってすべてのＵ．Ｌ．契約を満たせなくなった場合に、優先度の高い順にスケジューリングを行う。なお、優先度８０５は、サービスプロバイダとの契約に応じて定めても良いし、データセンタ側が顧客であるサービスプロバイダやサービスの優先度を判断して独自に定めても良い。

【００７８】

上記図３はＳＬＡの基本的なデータベースの構成であるが、これに加えてデータセンタ独自の項目を設定することも可能であり、例えば、提供するサービスの処理負荷などに応じた優先度などを設定できる。

【００７９】

したがって、ＳＬＡデータベース１１３（優先度情報保持手段）においては、提供するサービス（顧客、処理＝サービスフローの種類）に対応して、優先度、上限値（許容待ち時間）が定義されており、この定義は、図示しない入力手段を介してオペレータなどが設定、格納するもので、プリプロセッサ１０３及びスケジューラ１０４が、ＳＬＡデータベース１１３の優先度情報を参照し、スケジューラ１０４（実行優先度決定手段）は後述のように、サービスを識別する情報をキーとして、サービスプロバイダ名、サービス名を検索し、ＳＬＡデータベース１１３の優先度情報を読み込む。

【００８０】

４．メッセージディクショナリ

図４および図６を参照して、メッセージディクショナリ１１４の構成例を説明する。メッセージディクショナリ１１４は、メッセージ形式の、サービスプロバイダおよびサービス毎の定義を格納する。

【００８１】

トランザクションモニタ１２０の受け付けるメッセージは、固定部（図６の１００１、１００２）と可変部（図６の１００３）より構成される。固定部はトランザクションモニタ１２０に固有のメッセージフィールドであり、一方、可変部は、サービスプロバイダ、およびサービス毎に異なるメッセージのフィールドである。

【００８２】

図６のメッセージ構成に対応して、メッセージディクショナリ１１４も固定部定義｛図４（ａ）｝と可変部定義｛図４（ｂ）、（ｃ）、（ｄ）｝を有する。

【００８３】

図４（ａ）の例では、固定部定義は開始バイト（９０１）、長さ（９０２）、タイプ（９０３）のカラムを有する。メッセージの第０バイトから長さ３２バイト分のフィールドにサービスプロバイダ名称が、第３２バイトから長さ３２バイト分のフィールドにサービス名称が格納されることを示す。第６４バイト以降のフィールドは、可変部に相当する。

【００８４】

可変部定義は、可変部インデクス定義｛図４（ｂ）｝と、可変部フィールド定義｛図４（ｃ）、（ｄ）｝を組み合わせて行う。

【００８５】

可変部インデクス定義は、メッセージ固定部１００１、１００２に入れられたサービスプロバイダ名９０５およびサービス名９０６（サービスを識別する情報）をキーとして、可変部フィールド定義のインデクスを検索するためのテーブルである。図４（ｂ）では、例えば「サービスプロバイダＡ」の「サービスＡ１」のインデクスは「１」である。

【００８６】

これと同じテーブルインデクス（＝「１」）をもつ可変部フィールド定義｛図４（ｃ）｝が、「サービスＡ１」に関する定義となる。同様に、「サービスプロバイダＡ」の「サービスＡ２」のインデクスは「２」である。これと同じテーブルインデクスをもつ可変部フィールド定義｛図４（ｄ）｝が、「サービスＡ２」に関する定義となる。

【００８７】

各テーブルインデクスは、開始バイト、長さ、データタイプの列に応じたフィールドが設定されており、図４（ｃ）では、メッセージの第６４バイトから４バイト分のフィールドに口座番号が、第６８バイトから１２バイトのフィールドにタイムスタンプが、第８０バイトから８バイト分のフィールドに引き出し額が格納されることを示す。図４（ｄ）も同様であるが、第８０バイトから８バイト分のフィールドは現在残高に対応する。

【００８８】

以上のような定義により、あるメッセージがトランザクションモニタ１２０に入力された場合に、固定部１００１、１００２によってそのメッセージがどのサービスプロバイダのどのサービスに属するかが判別できる。また、可変部１００３によって該サービスに対するパラメータを指定することができる。

【００８９】

５．サービスフロー定義及びサービスフロー実行部

サービスフロー実行部１０７は、図５に示すように、メッセージを用いて個別の処理を連結することにより形成する。単一の目的を持った処理（処理ノード）を複数組み合わせることによって複雑な機能を実現することができる。

【００９０】

図５（ａ）では、サービスフローが６０１～６０５の５個の処理ノードからなる場合を示し、図中の矢印はノード間のメッセージの流れを示す。

【００９１】

ノード６０１はトランザクションモニタを経由して端末からのメッセージを受け取り、これを処理ノード６０２に渡す。処理ノード６０２は、メッセージを参照しつつ利用者によって定義された処理を行い、必要に応じてメッセージを変形してさらに下流のノード６０３、６０４、６０５にメッセージを渡す。

【００９２】

６０３はメッセージの変換ノードであり、出力先のサービスプログラムの形式に合わせて、メッセージ中のコード変換（例えばＥＢＣＤＩＣコードからＡＳＣＩＩコードへの変換）を行う。６０４、６０５は出力ノードであり、トランザク

ションモニタ経由で外部のサービスプログラムに対してメッセージを送出する。

【００９３】

図５（ｂ）には、サービスフローの定義情報１１５を示す。

【００９４】

６２０～６２４は、それぞれ６０１～６０５のノードに対する定義情報である。サービス名は、各ノードが属するサービスの名称を指定する。ノード名称には任意のノード名称を、フロー内で一意になるように指定する。ノード種類は、入力ノード、処理ノード、変換ノード、出力ノードなど、当該メッセージブローカシステムにて提供されるノードの種類から適当なものを選んで指定する。入力先、出力先には、当該ノードに対するメッセージの入出力先ノード名称を指定する。例えばノードＢ６０２はノードＡ６０１からメッセージを受け入れ、ノードＣ６０３およびノードＥ６０５にメッセージを出力する。処理ノード、変換ノードには、個別に処理内容を指定する。

【００９５】

処理内容の指定は、定義情報６２０～６２４の「モジュール」カラムに、対応する処理モジュールの名称を格納することで可能となる。入力・出力ノードのような定型処理を行うノードに対しては、予め固定されたモジュールを用いるため、名称の指定は必要ない。

【００９６】

このサービスフロー定義１１５とサービスフロー実行部１０７により、トランザクションモニタ上でサービスフローの実行を可能とすることにより、複数のサービスプログラムを組み合わせてより複雑なサービスを提供するメッセージブローカを構築することができる。

【００９７】

６．実行モジュールライブラリ

実行モジュールライブラリ１１６は、サービスフロー中の各サービスノードを実行するのに必要な実行モジュール群を格納する。各実行モジュールは例えばＵＮＩＸのファイル形式でディスクに格納することができる。ファイル名称をサービスフロー定義中に現れるモジュール名称と一致させることにより、サービスフ

ローから実行モジュールの検索が可能となる。

【００９８】

また、実行モジュールは実行時に動的にローディングが可能な形式、例えばＵＮＩＸのＤＬＬ（Ｄｙｎａｍｉｃ　Ｌｏａｄｉｎｇ　Ｌｉｂｒａｒｙ）形式で作成する。

【００９９】

７．リクエストキュー

リクエストキュー１１０、１１１は、トランザクションモニタ１２０に入力されたメッセージを入力順に格納するデータ構造である。

【０１００】

リクエストキュー１１０、１１１は、トランザクションモニタ１２０に登録されたサービスプロバイダのサービス毎に一つずつ作成する。リクエストキューの構造の例を図７に示す。

【０１０１】

図７において、リクエストキューは、キュー毎に一つ存在するリクエストヘッダ１１１４～１１１６と、そこからリスト構造でつながる複数のリクエスト構造体１１０１～１１０４よりなる。

【０１０２】

リクエストヘッダは、サービス情報１１１４、ＳＬＡ情報１１１５、後方チェイン１１１６、キューの先頭ポインタ１１１７およびキュー情報１１１８の各フィールドを有する。

【０１０３】

サービス情報１１１４はキューに割り当てられたサービスプロバイダおよびサービス名称を格納する。ＳＬＡ情報１１１５はＳＬＡデータベース１１３に格納されたＳＬＡ定義を、サービスプロバイダ名称およびサービス名称をキーとして検索することにより取得し、リクエストヘッダに格納する。

【０１０４】

後方チェイン１１１６は、キューが複数存在する場合に、リクエストヘッダをリスト構造にて連結するためのポインタを格納する。図中、１１３０～１１３２

は、複数のキューが後方ポインタを用いて連結された状態を示す。

【０１０５】

キューの先頭ポインタ１１１７は、キューの先頭リクエスト構造体（最初に作成された構造体）へのポインタを格納する。キュー情報１１１８は、キューにつながれたリクエストの状況を格納するフィールドであり、その使用法は後述する。

【０１０６】

各リクエスト構造体には、４つのフィールド１１１０～１１１３が存在する。タイムスタンプ１１１０は、リクエストの生成時刻を示すフィールドである。前方チェイン１１１３および後方チェイン１１１４は、リクエスト構造体を相互に連結してキューを形成するためのポインタを格納する。メッセージポインタ１１１５は、メッセージ本体の格納領域に対するポインタを格納する。

【０１０７】

１１０１～１１０４は、前方チェイン、後方チェインを用いてリクエスト構造体がキューを構成した状態の例を示す。１１２０～１１２３は、各リクエストに対応するメッセージの格納領域であり、対応するリクエスト構造体よりポイントされる。

【０１０８】

８．プリプロセッサ

プリプロセッサ１０３は、トランザクションモニタ１２０に入力されたメッセージをメッセージディクショナリ１１４の内容と突き合わせ、当該メッセージがどのサービスプロバイダのどのサービスに対応するかを解析して、適切なリクエストキュー１１０、１１１に格納する。

【０１０９】

図８にプリプロセッサの動作フローの例を示す。

【０１１０】

プリプロセッサ１０３は起動時に、メッセージディクショナリ１１４からメッセージ固定部（図６中１００１、１００２）の情報を読み込む（１２０１）。

【０１１１】

次に、トランザクションモニタ１２０のサービス終了まで、クライアントからのメッセージを受け付けるループ１２０２に入り、メッセージ入力待ち状態となる（１２０３）。サービス終了は、特殊なメッセージを受け付けることによってループ１２０２から抜けても良いし、割り込み処理によってループを終了しても良い。

【０１１２】

メッセージの入力待ちは、例えばＵＮＩＸのａｃｃｅｐｔシステムコールなどにより実現できる。メッセージが入力されると、すでに１２０１にて読み込んだメッセージ固定部情報を用いて、メッセージに対応するサービスプロバイダ名、サービス名を切り出す（１２０４）。

【０１１３】

次に、リクエストヘッダを順次サーチして（１２０５）、取得したサービスプロバイダ名、サービス名に該当するキューを検索し（１２０６）、そのキューに入力されたメッセージ（電文）を登録する。メッセージの登録は、メッセージとサービスプロバイダ名、サービス名を含むメッセージ構造体（図７中１１１０～１１１３）を新たに作成し、キュー構造（図７中１１０１から１１０４）の最後尾に、作成した構造体をポインタ操作により挿入することにより実現できる。

【０１１４】

９．キューイング状況検出部

キューイング状況検出部１１２は、各リクエストキュー１１０、１１１の状況を監視し、スケジューラ１０４がスケジュールすべきリクエストを選択するとともに、適当な量のリソースを各サービスに配分するために必要な情報を抽出する。

【０１１５】

キューイング状況検出部１１２は、トランザクションモニタ１２０またはサーバのオペレーティングシステムによって一定時間毎に起動され、所定の処理を行う。一定時間毎の起動には、例えばＵＮＩＸオペレーティングシステムのｓｉｇａｌａｒｍシステムコールを用いることができる。

【０１１６】

毎回の起動によって実行される処理フローの例を図９に示す。

【０１１７】

各リクエストヘッダについて（１３０１）、そこからポイントされるリクエスト構造体をスキャンし（１３０２）、キュー中のリクエスト数をカウントアップする（１３０３）と同時に、当該キューのリクエスト構造体に格納されたタイムスタンプのうち最も古いものを求める。

【０１１８】

このようにして取得したリクエスト数と、最も古いタイムスタンプとを、リクエストヘッダのキュー情報フィールド（図７の１１１８）に格納する。

【０１１９】

１０．スケジューラ

スケジューラ１０４は、キューイング状況検出部１１２によって抽出された情報に基づき、リクエストのスケジューリングを実行する。

【０１２０】

スケジューリングは、ＳＬＡのクラスがＵ．Ｌ．（アッパーリミット契約）であるリクエストを優先し、余裕がある場合にＢ．Ｅ．クラス（ベストエフォート契約）のリクエストをスケジューリングする。各々のクラスのうちでは、タイムスタンプがもっとも古いリクエストから優先的に処理を行う。

【０１２１】

図１０に、スケジューリングすべきリクエストの選択処理フローの具体例を示す。

【０１２２】

まず、各一時変数を初期化する（１４０１）。ここで、ＴｕｌはＵ．Ｌ．クラスサービスプロバイダに属するリクエストのタイムスタンプを格納する一時変数、Ｔｂｅは同様にＢ．Ｅ．クラスのリクエストのタイムスタンプを格納する一時変数である。Ｐｕｌ、ＰｂｅはそれぞれＵ．Ｌ．クラス、Ｂ．Ｅ．クラスのリクエストに対するポインタを格納する一時変数である。

【０１２３】

次に、リクエストヘッダリスト（図７中１１３０～１１３２）に格納された各

ヘッダについて（１４０２）、ヘッダ中のＳＬＡ情報（図７中１１１５）を参照して、当該ヘッダがＵ．Ｌ．クラスかＢ．Ｅ．クラスかを判別する（１４０３）。

【０１２４】

Ｕ．Ｌ．クラスであった場合には、Ｔｕｌに格納されたこれまでのタイムスタンプの最小値と、当該リクエストヘッダに格納されたキュー情報（図７中１１１８）より得られるキューの最古タイムスタンプとを比較する（１４０４）。当該タイムスタンプの方が古かった（小さかった）場合には、一時変数Ｔｕｌを入れ替えるとともに（１４０５）、当該リクエストヘッダへのポインタを一時変数Ｐｕｌに格納する。

【０１２５】

一方、上記判別１４０３において当該ヘッダがＢ．Ｅ．クラスに属すると判定された場合には、一時変数ＴｂｅとＰｂｅとを用いて同様の処理を行う（１４０７～１４０９）。以上の処理によって、Ｕ．Ｌ．クラス、Ｂ．Ｅ．クラスのそれぞれについて最古のタイムスタンプと、それに対応するリクエストヘッダを求めることができる。

【０１２６】

次いで、Ｕ．Ｌ．クラス、Ｂ．Ｅ．クラスのいずれのリクエストをスケジュールすべきかを決定する。

【０１２７】

まず、ＰｕｌかＰｂｅのいずれかがＮＵＬＬであった場合には、ＮＵＬＬでない方のリクエストをスケジューリングする（１４１０～１４１２）。

【０１２８】

双方ともＮＵＬＬでなかった場合には、以下の式１）を評価する（１４１３）。

Ｔｕｌ＜（（現在時刻－上限値）＋ｅ）　　・・・（１）

ここに、現在時刻は本処理を実行中の時刻である。上限値はＳＬＡデータベース１１３に定義されたＳＬＡ契約における当該サービスの上限時間（図３中８０４）であり、リクエストヘッダ（図７中１１１５）中のＳＡＬ情報を参照して得ら

れる。また、ｅは本トランザクションモニタ１２０の運用者により決定されるオフセットである。

【０１２９】

上記（１）式は、Ｕ．Ｌ．クラスの最も古いタイムスタンプを持つリクエストが、ＳＬＡ契約における処理の後れの上限値からある時間幅（ｅ）以内に存在するか否かを調べることに相当する。存在する場合には、Ｕ．Ｌ．クラスを優先し、このリクエストをスケジュール対象とする（１４１４）。一方、時間幅（ｅ）以内に存在しない場合には、Ｕ．Ｌ．クラスの処理に余裕があるので、Ｂ．Ｅ．クラスのリクエストのうちもっとも古いタイムスタンプをもつものをスケジューリングする（１４１５）。

【０１３０】

１１．ダイナミックローダ

ダイナミックローダ１０５は、スケジューラ１０４によるリクエストのスケジューリング結果を受けて、プロセスの起動と実行モジュールのローディングを行う。

【０１３１】

ダイナミックローダ１０５内部には、サービスフロー実行部１０７にて起動されたプロセスの状態を管理するためのプロセス管理情報を有する。

【０１３２】

図１１を用いて、プロセス管理情報の構成例を説明する。

【０１３３】

プロセス管理情報は、各サービスに対してどのプロセスが対応しており、その各プロセスに対してどの実行モジュールがローディングされているかを管理する。

【０１３４】

サービス構造体は１５０１～１５０３のフィールドを有し、サービス名称を格納する。この構造体を１５２１、１５２２のように連結することにより、サービス対応のリスト構造（サービスチェイン）を作成する。

【０１３５】

各サービス構造体１５２１、１５２２からは、プロセス構造体１５３１、１５３２がプロセスポインタ１５０２によってポイントされる。

【０１３６】

プロセス構造体１５３１、１５３２は、４つのフィールドを有し、プロセスＩＤ１５０４、実行モジュールへのポインタ１５０３、該プロセスが使用中か否かを示すフラグ１５０４および後方ポインタ１５０５を格納する。

【０１３７】

本構造体を１５３１、１５３２のように連結してリスト構造（プロセスチェイン）を形成する。さらに、プロセス構造体１５３１、１５３２からは実行モジュール構造体１５４１～１５４３及び１５５１～１５５３がポイントされる。

【０１３８】

実行モジュール構造体は、モジュール名称１５０８、後方ポインタ１５０９および実行モジュールのバージョンを示すカウンタ情報１５１０を格納する。実行モジュール構造体を１５４１～１５４３（あるいは１５５１～１５５３）のように連結してリスト構造（モジュールチェイン）を形成する。

【０１３９】

次に、図１２を用いてダイナミックローダの処理フローを説明する。

【０１４０】

まず、プロセス管理情報中のサービスチェイン（図１１中１５２１、１５２２）をたどり、スケジューリング対象サービスがチェイン中に存在するかを調べる（１６０２）。

【０１４１】

存在する場合には、そのサービスを処理するための少なくとも一つのプロセスがすでに起動されているので、プロセスチェイン（図１１中１５３１、１５３２）をたどって、そのうち未使用のものをサーチする（１６０３、１６０４）。

【０１４２】

未使用のプロセスが存在する場合には、実行モジュール管理部１０６中の実行モジュールテーブル１７００を共有ロックし、そのプロセスを構成するモジュールチェイン（図１１中１５４１～１５４３）をたどり、各実行モジュールが前回

のローディング以降変更または更新されたかを調べる（１６０７、１６０８）。

【０１４３】

実行モジュール管理部１０６、実行モジュール管理テーブル１７００の詳細については後述する。変更が検出された場合には、該モジュールをローディングする（１６０９）。

【０１４４】

スケジューリング対象のサービスがチェイン中に存在しない場合（１６０５）、またはプロセスチェイン中に未使用のプロセスが無い場合（１６０６）には、新たにプロセスを起動して、必要な実行モジュールのローディングを行う（１６２１）。

【０１４５】

まずプロセス起動を行って、そのＩＤをプロセスチェインに登録し（１６２２）、サービスフロー定義１１５中のサービスフロー定義テーブルの各カラムについて（１６２３）、各モジュールが現在着目するサービスに属するか否かを判定し（１６２４）、属する場合にはローディングを行う（１６２５）。なお、プロセスのＩＤとしては、たとえばＵＮＩＸオペレーティングシステムで付されるｐｉｄを利用すれば良い。

【０１４６】

１２．実行モジュール管理部

実行モジュール管理部１０６は、実行モジュールライブラリ１１６中の実行モジュールの追加、更新、削除を管理する。実行モジュールの状態を保持するデータ構造として、実行モジュール状態テーブルを有する。

【０１４７】

実行モジュール状態テーブルの例を図１３に示す。

【０１４８】

状態テーブル１７００のカラム１７０１～１７０５は実行モジュールライブラリ１１６に格納された各実行モジュールに対応する。各々の実行モジュールに対し、実行モジュール名称および更新カウンタ情報（識別子）を格納する。

【０１４９】

更新カウンタは当該実行モジュールの更新数を表す整数であり、モジュール新規登録時には１を格納し、モジュールが更新される毎に１ずつインクリメントする。

【０１５０】

さらに、実行モジュール状態テーブル１７００にはロックフィールド１７１０が付随する。本フィールドはテーブルのロック状況を格納し、Ｎ（ロック無し）、Ｓ（共有ロック中）、Ｅ（排他ロック中）の３通りの値をとる。

【０１５１】

ここで、ダイナミックローダ１０５が本状態テーブルを用いて実行モジュールの変更状態を検出するステップ（図１２中１６０８）について、図１４を用いて詳細を説明する。

【０１５２】

まず、ダイナミックローダ１０５は、実行モジュール状態テーブル１７００のロックフィールドを共有ロックし（図１２の１６０６）、ループ１６０７にて現在着目している実行モジュールの名称を対応するモジュール構造体（例えば１５４１）から取得する（１８０１）。

【０１５３】

次に、この名称を用いて実行モジュール状態テーブルをルックアップし（１８０２）、対応する更新カウンタを取得する（１８０３）。さらに、取得したカウンタの値とモジュール構造体（例えば１５４１）中のバージョンカウンタ（図１１の１５１０）の値とを比較する（１８０４）。

【０１５４】

更新カウンタとバージョンカウンタの値が同じであった場合には、前回ローディング時から実行モジュールに変更はなかったと判断し、実行モジュールの再ローディングは行わない。

【０１５５】

一方、更新カウンタがバージョンカウンタよりも大きかった場合には、前回ローディング時以降に実行モジュールが変更されたとみなし、実行モジュールを再ローディングするとともに、バージョンカウンタに更新カウンタの値を代入する

【０１５６】

一方、実行モジュールライブラリにモジュールを更新する場合の、実行モジュール管理部１０６の処理フローを図１５を用いて説明する。

【０１５７】

まず、実行モジュール状態テーブル１７００のロックフィールド１７１０を排他ロック（「Ｓ」）する。

【０１５８】

続いて、更新対象となる実行モジュールライブラリの名称を、トランザクションモニタ１２０より取得する。

【０１５９】

この名称は、例えば管理者がトランザクションモニタの管理コンソール（図２中７２０）から入力する情報から取得することができる。続いて、実行モジュール状態テーブル１７００をサーチし、更新対象モジュール名称と同じ名称を持つカラムを見出し（１９０４）、そのカラムの更新カウンタをインクリメントする（１９０５）。最後に、実行モジュール状態テーブル１７００の排他ロックを解除（「Ｎ」）する。

【０１６０】

１３．作用

以上のような構成により、リクエストキュー１１０、１１１を、トランザクションモニタ１２０に登録されたサービスプロバイダのサービス毎に一つずつ設けるのに加え、入力されたメッセージをメッセージディクショナリ１１４の内容に基づいて適切なリクエストキュー１１０、１１１に送るプリプロセッサ１０３と、リクエストキュー１１０、１１１の状況を監視して、スケジューラ１０４がスケジュールすべきリクエストを選択するキューイング状況検出部１１２と、ＳＬＡデータベース１１３に格納された複数のサービスプロバイダ（顧客）とのサービスの優先度を表す情報（サービスレベルに関する契約情報）をキーとして、スケジューラ１０４がリクエストを制御することにより、複数の顧客でひとつのトランザクションモニタ１２０（またはメッセージブローカ）を共有しながらも、

所定の優先度またはリソースの状態に応じて最適なリソースの割り当てが行われるため、適切なスループットを保証することができるのである。

【０１６１】

したがって、複数のサービスプロバイダのシステムをアウトソーシング形態で受託して、計算機資源を集中管理するデータセンタなどに適用すれば、より少ない計算機資源（リソース）で、より多くのオンライントランザクションを、顧客との保証契約に応じたスループットを確保しながらリアルタイムで処理することが可能となり、複数の顧客の業務を集中して処理するデータセンタの信頼性と処理能力を向上させることができるのである。

【０１６２】

また、複数のサービスプロバイダのサービス毎に、必要なプロセスを実行するダイナミックローダ１０５は、各プロセスを構成する実行モジュールが更新されたか否かを検出する実行モジュール管理部１０６に基づいて、更新があったモジュールを一括してローディングした後にトランザクション処理を行うようにしたため、実行モジュールのルーティング停止や予備のプロセスを設けることなく、トランザクションモニタ１２０稼働中にサービスの変更を随時行うことが可能となり、計算機資源の有効利用を維持しながらも顧客の業務ロジックの追加、変更を容易に行って、柔軟性、可用性に富んだトランザクションモニタ１２０またはメッセージブローカを構築することができ、システムの運用を簡易にすることができるのである。

【０１６３】

図１６、図１７は、第２の実施形態を示す。

【０１６４】

前記第１の実施例においては、サービスフロー実行部１０７の有するプロセス数に十分な余裕があり、スケジューラは必ずリクエストのスケジューリングが可能であることを想定した。

【０１６５】

これに対し、本実施形態では、計算機資源の制約からプロセス数が必ずしも十分に確保できず、サービス間でプロセスのトレードオフが必要となる場合を想定

し、より一般的な場合を想定する。

【０１６６】

本実施形態では、スケジューラ１０４に代えて、プロセス管理部２００１を設ける。なお、その他は前記第１実施形態と同様である。

【０１６７】

プロセス管理部２００１の役割は、各サービスのリクエストキュー１１０、１１１の状態とＳＬＡ契約とから、その処理に必要とされるプロセス数を推定し、ダイナミックローダ１０５を制御することにある。

【０１６８】

各プロセスは、現在処理中のトランザクションが修了するとリクエスト受け付け状態に入り、対応するリクエストキュー１１０、１１１からリクエストを取り出して、次のトランザクションの処理を行う。

【０１６９】

プロセス管理部２００１における処理の流れを、図１７に基づいて説明する。

【０１７０】

プロセス管理部２００１では、システム開始時にＳＬＡデータベース１１３から各サービスに関する上記図３のＳＬＡ情報を取得する。プロセス管理部２００１は、システム稼働中は定期的にキューとプロセスの状況を監視し（２１０３）、以下の処理を行う。

【０１７１】

まず、キューイング状況検出部１１２から各サービスについてのキュー情報を取得する（２１０２）。キュー情報は、滞留リクエストの個数および最古のタイムスタンプであり、上記図９にて説明した通り、キューイング状況検出部１１２で抽出されたリクエストヘッダのキュー情報フィールド１１１８を参照して取得できる。

【０１７２】

次いで、各サービスについて（２１０４）、該サービスに対応するトランザクションの開始時刻、終了時刻、プロセス数をサービスフロー実行部から取得し、該サービスに対応するトランザクションのスループットを計算する。通常一つの

サービスに対応するプロセスは複数あるので（図１６中１０８、１０９）、当該サービスに対するトータルスループットは、各プロセスが処理するトランザクションに要する時間の逆数の総和をとることにより求められる。

【０１７３】

一方、取得したキュー情報から、前回と今回のキュー長（キュー中の滞留リクエスト数）の差分を求め、リクエスト到着頻度を計算する。リクエスト到着頻度は、キュー長の差分を、時間間隔で除することにより計算できる。

【０１７４】

あるいは、各トランザクションの開始時刻と終了時刻の差を取得し、その逆数に、各サービスに割り当てたプロセス数を乗ずることでスループットを求めてもよい。

【０１７５】

このようにして求めたトータルスループットとリクエスト到着頻度を比較することにより、該サービスに対するスループットの充足度を推定することができる。

【０１７６】

すなわち、トータルスループットがリクエスト到着頻度より大きければキューの長さは時間とともに減少し、小さければ増大する。ここではスループット充足度を、トータルスループットをリクエスト到着頻度で除することにより求める（２１０９）。

【０１７７】

以上で各サービスに対するスループットの充足度が求まったので、次に、各サービスに対するプロセス数を変更し、スループットが各サービスに対して最適に配分されるように制御する。ＳＬＡ契約で優先度の高いサービス順に（２１０８）、スループット充足度を１以上にするために必要な新プロセス数を計算する。新プロセス数が現プロセス数より大きい場合には、新たに差分のプロセスを起動し、ダイナミックローダ１０５を通して必要な実行モジュールのローディングを行ってそれらを受信待ち状態におく（２１１２）。

【０１７８】

トランザクションモニタ全体でプロセス数に制限があり、必要なプロセスをすべて起動できない場合には、可能な範囲で最大のプロセス数を起動する（２１１３）。一方、スループット充足度に余裕がある場合には、余裕分のプロセスを停止してシステムリソースを解放する（２１１４）。

【０１７９】

このようなスケジューリングによれば、ＳＬＡ契約で優先度の高いサービスに優先的にプロセスが配分され、契約が満たされる確立を高めると同時に、スループット充足度に余裕のある場合には優先度の低いサービスに資源を回すことが可能となる。

【０１８０】

したがって、計算機資源の制約からプロセス数が必ずしも十分に確保できない状態においても、ＳＬＡ契約に応じたスループットを確保できるように、計算機リソースの制御を行うことができ、信頼性を向上させることができる。

【０１８１】

なお、リクエスト到着頻度に変動が大きい場合など、図１７の処理ではプロセスの起動、停止が頻発する可能性がある。一度起動したプロセスは一定期間停止させないなど、履歴を考慮した制御を行うことにより、プロセス数の過度の変動を抑止することができる。

【０１８２】

また、図１７の処理では、優先度の高いリクエストが多数投入された場合に、優先度の低いリクエストの実行が滞る可能性がある。このような場合には、各サービスに対する最小プロセス数を予め決めておき、これを下回らない範囲でプロセス数の増減を行えば良い。

【０１８３】

さらに、上記第２実施形態の特徴的な部分としては、１つまたは複数のサービスを提供し、該提供する各サービスには１つまたは複数のクライアントが接続可能なトランザクション処理であって、該クライアントから前記提供するサービスに対して送付された処理要求を、提供するサービス毎に順序付けて格納するキューイング手段（１１０、１１１）と、キューイング手段に格納した処理要求の滞

留情報を取得する滞留情報取得手段（キューイング状況検出部１１２）と、前記各サービスに対するトランザクションのスループットを検出するスループット検出手段及び前記各サービスに対してトランザクションの処理プロセスを割り当てるプロセス割り当て手段（プロセス管理部２００１）と、を有し、このプロセス割り当て手段は、前記各サービスに対して、取得した処理要求滞留情報と検出したトランザクションスループットとを参照してプロセスの割り当てを決定するプログラムという点である。

【０１８４】

より詳しくは、プロセスの割り当てを、処理要求滞留情報から算出した単位時間における処理要求到着頻度と、トランザクションスループットとの大小を比較し、処理要求到着頻度がトランザクションスループットより大きい場合には割り当てるプロセス数を増加させ、一方、処理要求到着頻度がトランザクションスループットより小さい場合には割り当てるプロセス数を減少させるというプログラムという点である。

【図面の簡単な説明】

【図１】

本発明実施例の全体図。

【図２】

本発明のハードウェア構成図。

【図３】

ＳＬＡデータベースの説明図。

【図４】

メッセージディクショナリの説明図。

【図５】

サービスフローの説明図で、（ａ）はノードとサービスプログラムの関係を示し、（Ｂ）はサービス名に対応したノード名、ノード種類、入力先、出力先、モジュールの関係を示す。

【図６】

メッセージの説明図。

【図７】

リクエストキューの構成説明図。

【図８】

リクエストキューの動作説明図。

【図９】

キューイング状況検出部の動作説明図。

【図１０】

スケジューラの動作説明図。

【図１１】

プロセス管理情報の構成説明図。

【図１２】

ダイナミックローダの動作説明図。

【図１３】

実行モジュール状態テーブルの構成説明図。

【図１４】

ダイナミックローダの動作詳細説明図。

【図１５】

実行モジュール管理部の動作説明図。

【図１６】

第２の実施形態を示す構成図。

【図１７】

プロセス管理部の動作説明図。

【図１８】

従来のトランザクション処理システムの説明図。

【図１９】

従来のプロセス数制御の説明図で、（Ａ）はリクエストがひとつの場合、（Ｂ）はリクエストが溜まった場合、（Ｃ）はリクエストが減少した場合を示す。

【図２０】

従来の優先度制御の説明図。

【図２１】

従来のメッセージブローカの説明図。

【符号の説明】

１０１、１０２　クライアントプログラム

１０３　プリプロセッサ

１０４　スケジューラ

１０５　ダイナミックローダ

１０６　実行モジュール管理部

１０７　サービスフロー管理部

１０８、１０９　サービスフロー

１１０、１１１　リクエストキュー

１１２　キューイング状況検出部

１１３　ＳＬＡデータベース

１１４　メッセージディクショナリ

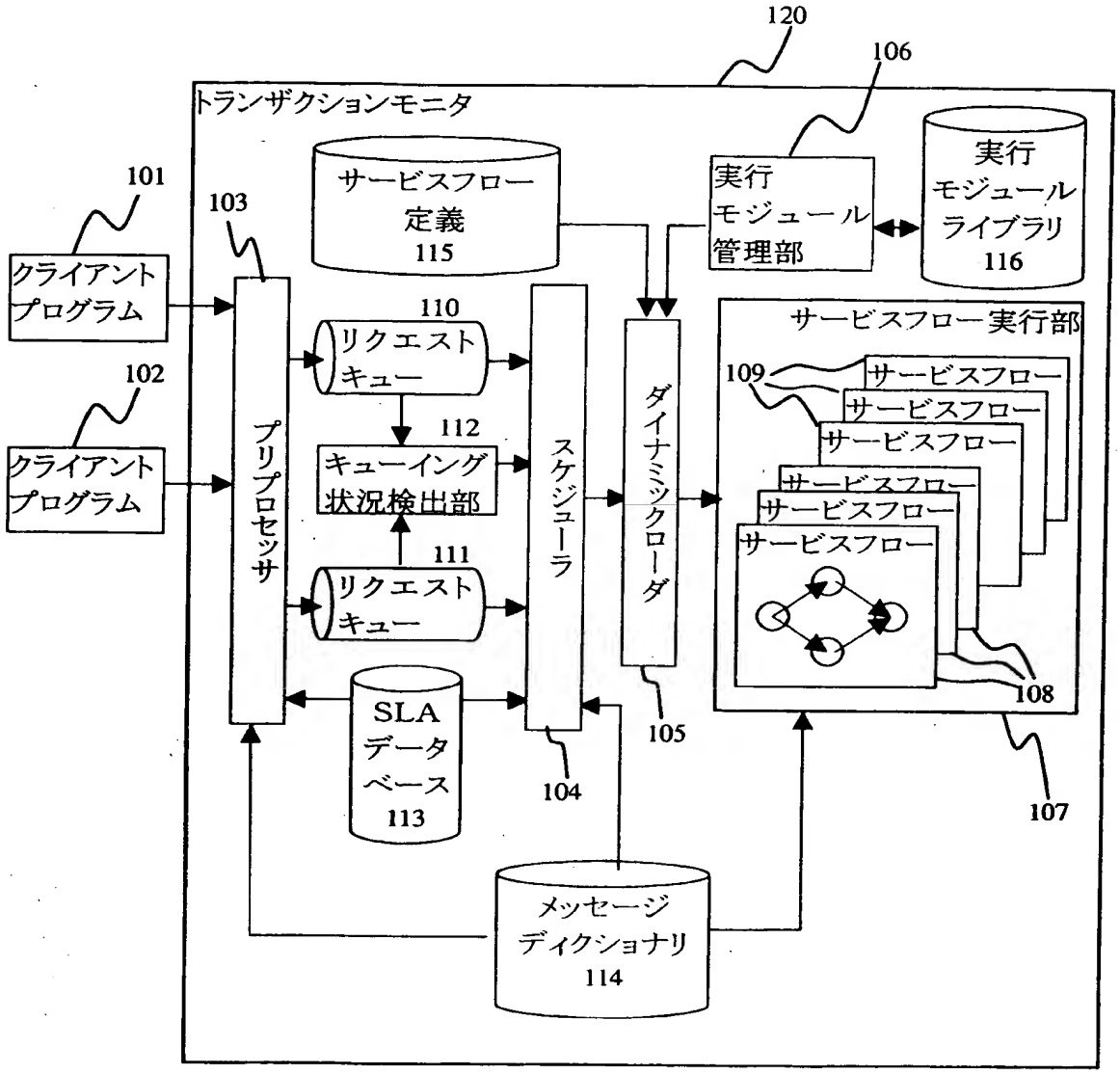
１１５　サービスフロー定義

１１６　実行モジュールライブラリ

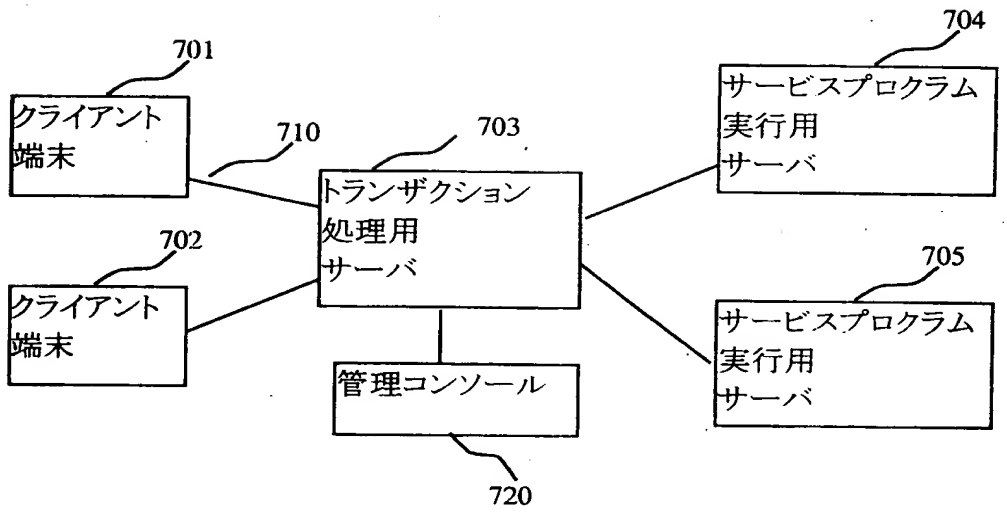
１２０　トランザクションモニタ

【書類名】　　図面

【図１】


120
106
トランザクションモニタ
101
103
サービスフロー
定義
115
実行
モジュール
管理部
実行
モジュール
ライブラリ
116
クライアント
プログラム
110
リクエスト
キュー
サービスフロー実行部
102
109
サービスフロー
サービスフロー
サービスフロー
サービスフロー
サービスフロー
サービスフロー
クライアント
プログラム
プリプロセッサ
112
キューイング
状況検出部
スケジューラ
ダイナミック
ローダ
111
リクエスト
キュー
108
SLA
データ
ベース
113
105
104
107
メッセージ
ディクショナリ
114

【図２】


701
クライアント
端末
710
703
トランザクション
処理用
サーバ
704
サービスプログラム
実行用
サーバ
702
クライアント
端末
705
サービスプログラム
実行用
サーバ
管理コンソール
720

【図３】

801 サービスプロバイダ名称	802 サービス名称	803 クラス	804 上限値	805 優先度
プロバイダA	サービスA1	U. L.	10	2
プロバイダA	サービスA2	B. E.		
プロバイダB	サービスB1	U. L.	20	1
プロバイダB	サービスB2	B. E.		
プロバイダB	サービスB3	U. L.	10	3

【図４】

(a)

901 開始バイト	902 長さ	903 タイプ	
0	32	文字列	サービスプロバイダ名称
32	32	文字列	サービス名称
64	-	-	可変部

(b)

905 サービスプロバイダ名称	906 サービス名称	907 テーブルインデクス
サービスプロバイダA	サービスA1	1
サービスプロバイダA	サービスA2	2
サービスプロバイダB	サービスB1	3

(c)

908 1

909 開始バイト	910 長さ	911 タイプ	
64	4	整数	口座番号
68	12	文字列	タイムスタンプ
80	8	整数	引き出し額

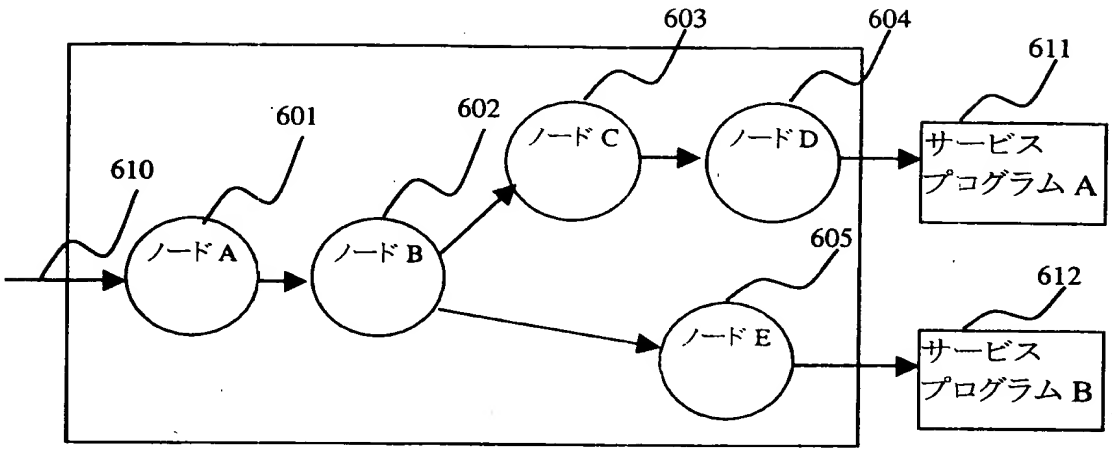
(d)

912 2

913 開始バイト	914 長さ	915 タイプ	
64	4	整数	口座番号
68	12	文字列	タイムスタンプ
80	8	整数	現在残高

【図５】

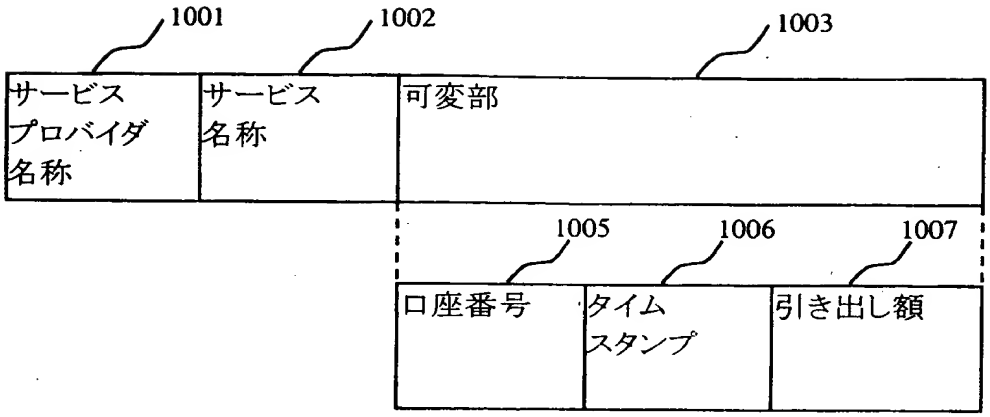
(a)



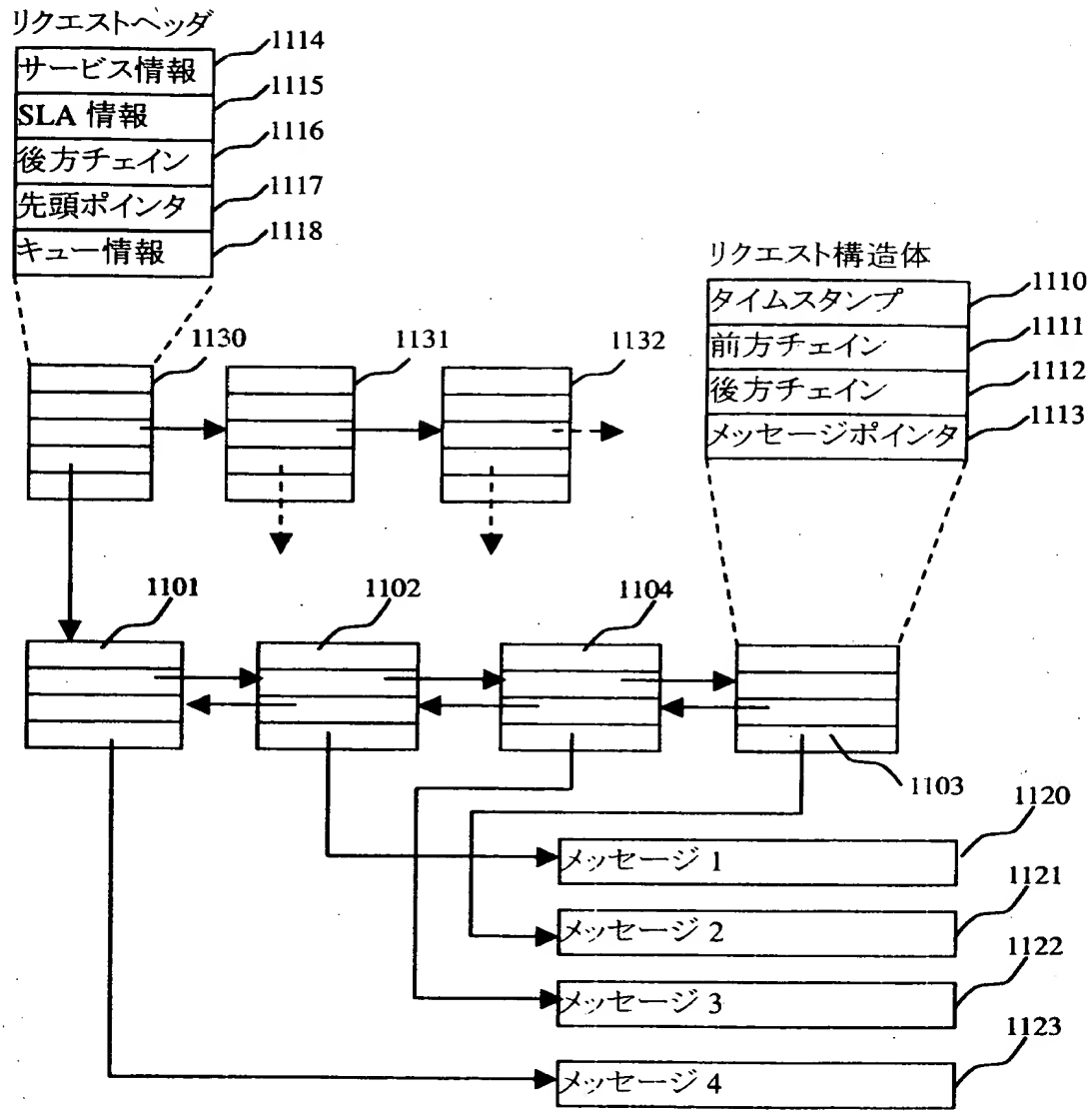
(b)

	620	621	622	623	624
サービス名	サービスA	サービスA	サービスA	サービスA	サービスA
ノード名称	ノードA	ノードB	ノードC	ノードD	ノードE
ノード種類	入力ノード	処理ノード	変換ノード	出力ノード	出力ノード
入力先		ノードA	ノードB	ノードC	ノードB
出力先1	ノードB	ノードC	ノードD		
出力先2		ノードE			
モジュール	"IN1"	"AG"	"TR"	"O1"	"O2"

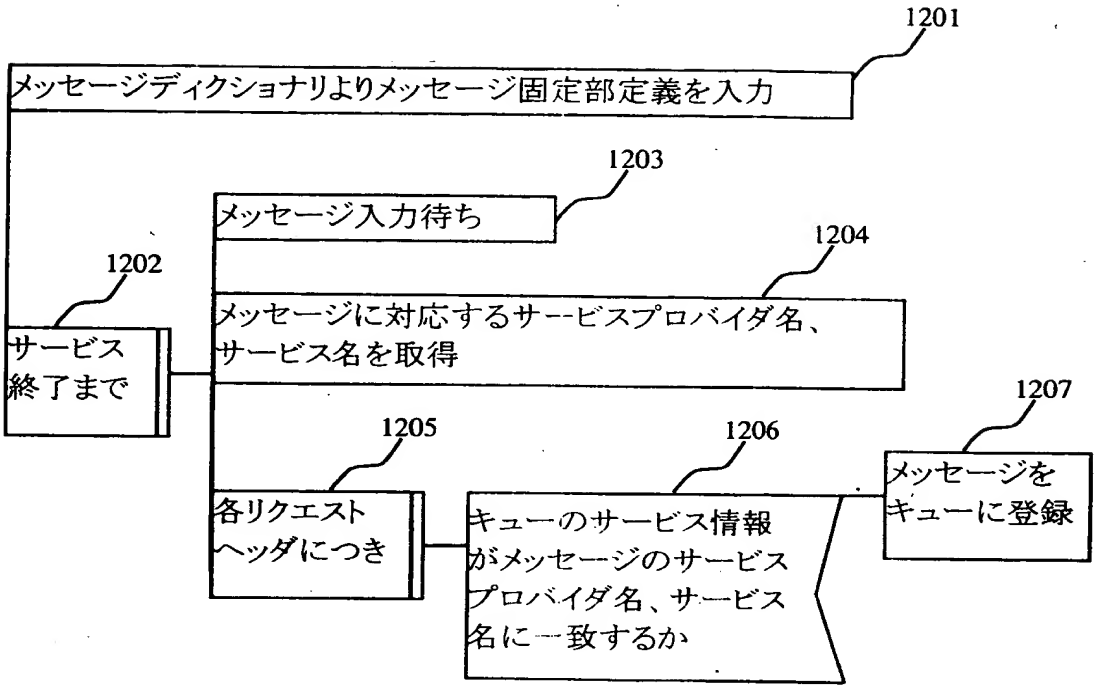
【図６】


1001
1002
1003
サービス
プロバイダ
名称
サービス
名称
可変部
1005
1006
1007
口座番号
タイム
スタンプ
引き出し額

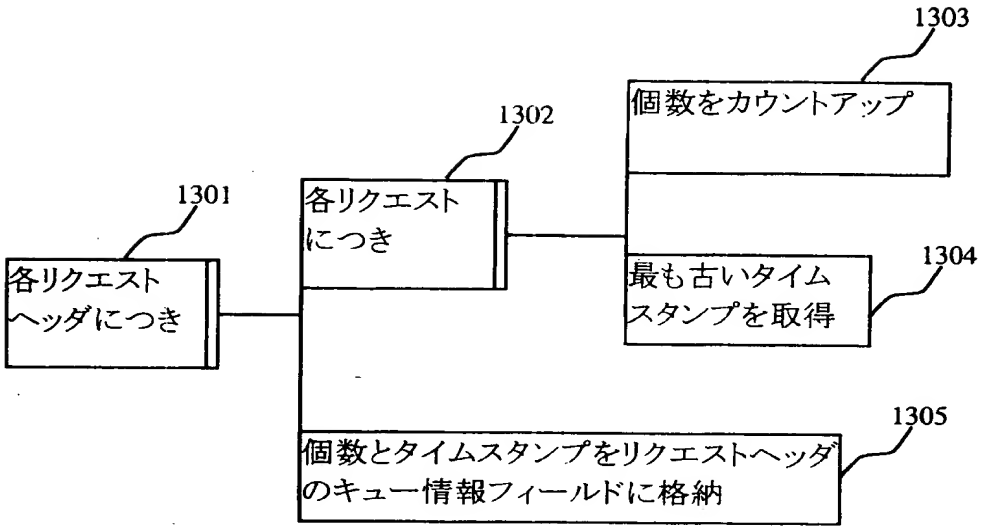
【図７】


リクエストヘッダ
サービス情報
SLA 情報
後方チェイン
先頭ポインタ
キュー情報
1114
1115
1116
1117
1118
1130
1131
1132
リクエスト構造体
タイムスタンプ
前方チェイン
後方チェイン
メッセージポインタ
1110
1111
1112
1113
1101
1102
1104
1103
メッセージ 1
メッセージ 2
メッセージ 3
メッセージ 4
1120
1121
1122
1123

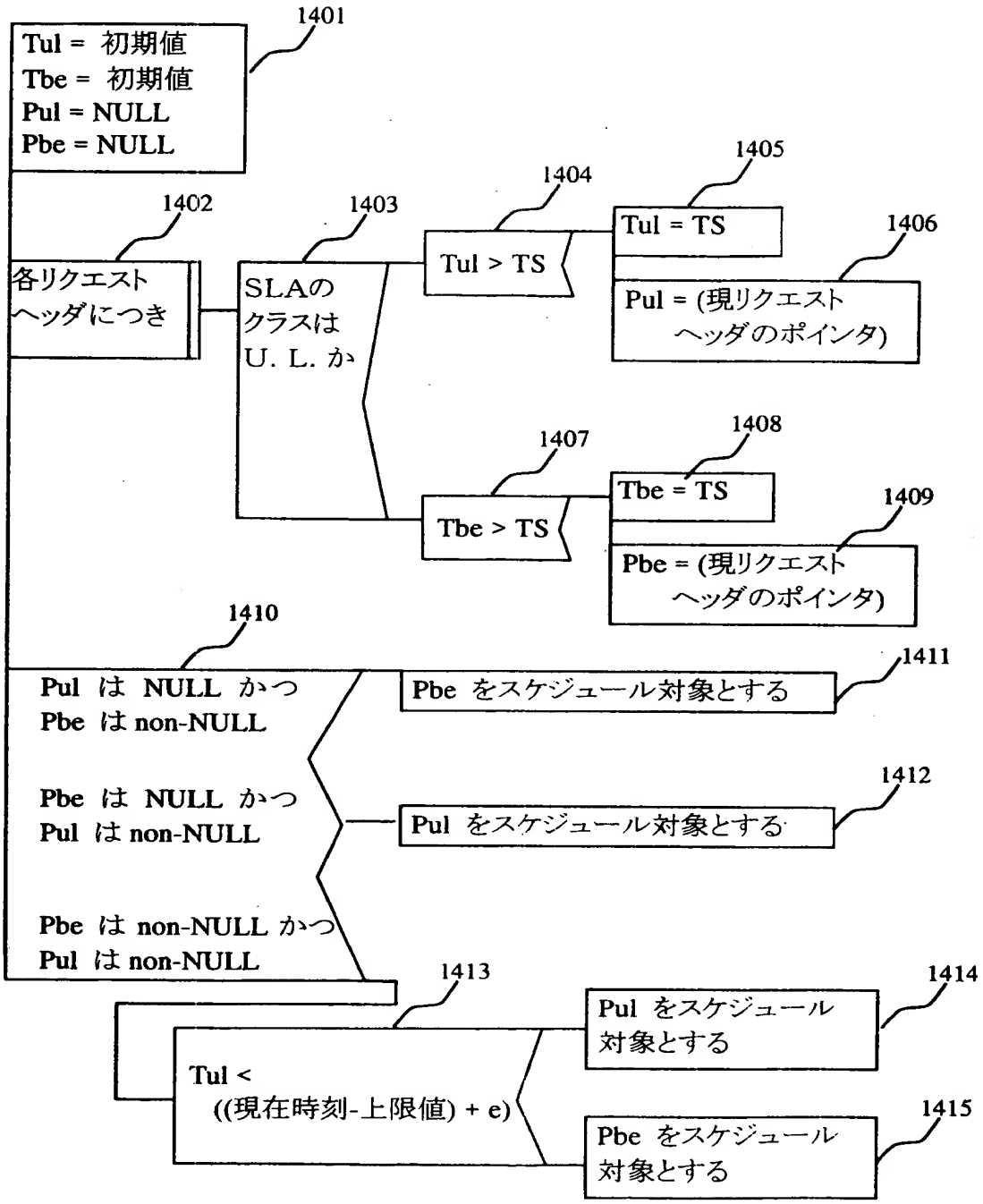
【図８】


1201
メッセージディクショナリよりメッセージ固定部定義を入力
1203
メッセージ入力待ち
1204
1202
メッセージに対応するサービスプロバイダ名、
サービス名を取得
サービス
終了まで
1207
1205
1206
メッセージを
キューに登録
各リクエスト
ヘッダにつき
キューのサービス情報
がメッセージのサービス
プロバイダ名、サービス
名に一致するか

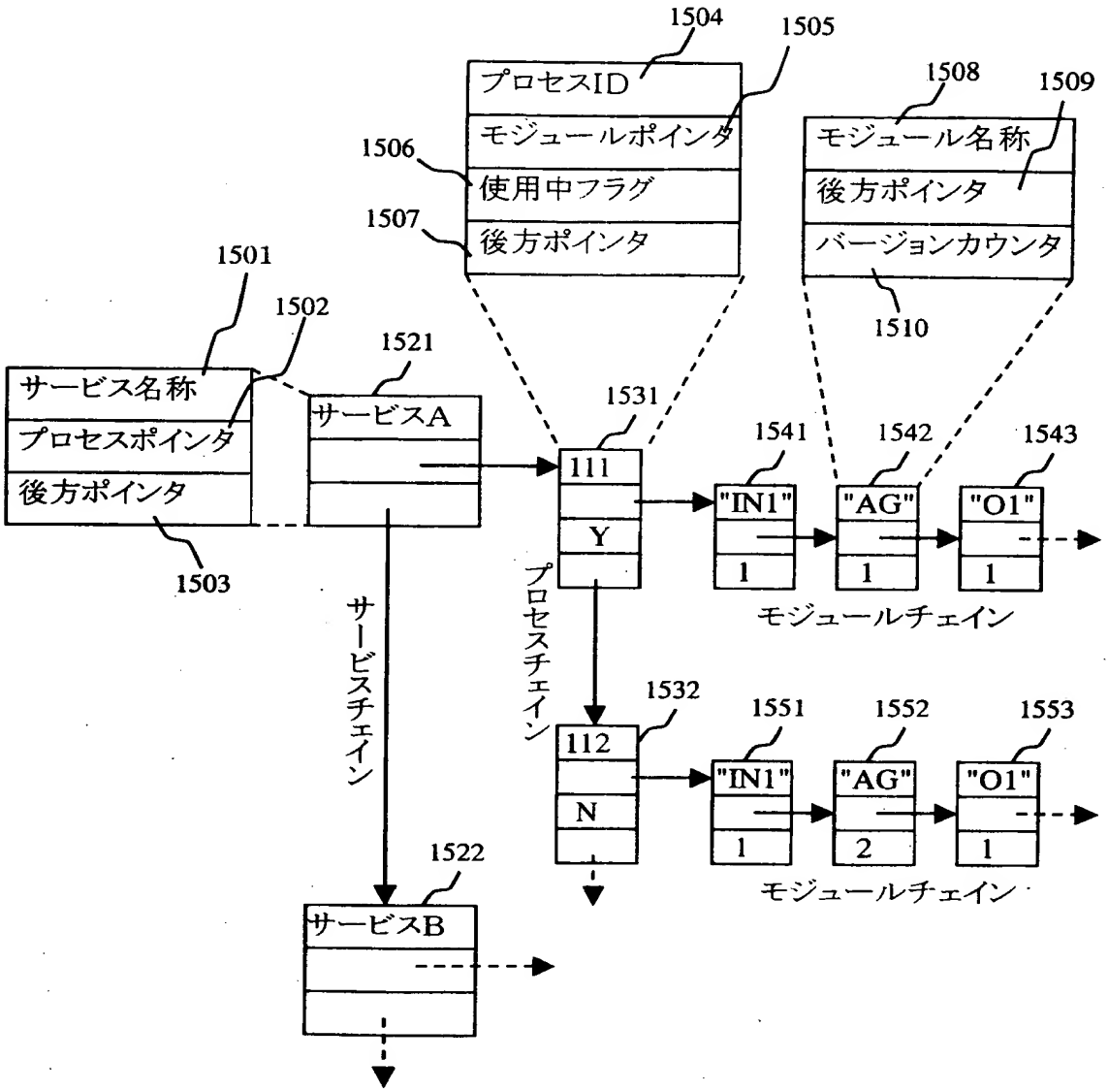
【図９】


1303
個数をカウントアップ
1302
各リクエスト
につき
1301
各リクエスト
ヘッダにつき
1304
最も古いタイム
スタンプを取得
1305
個数とタイムスタンプをリクエストヘッダ
のキュー情報フィールドに格納

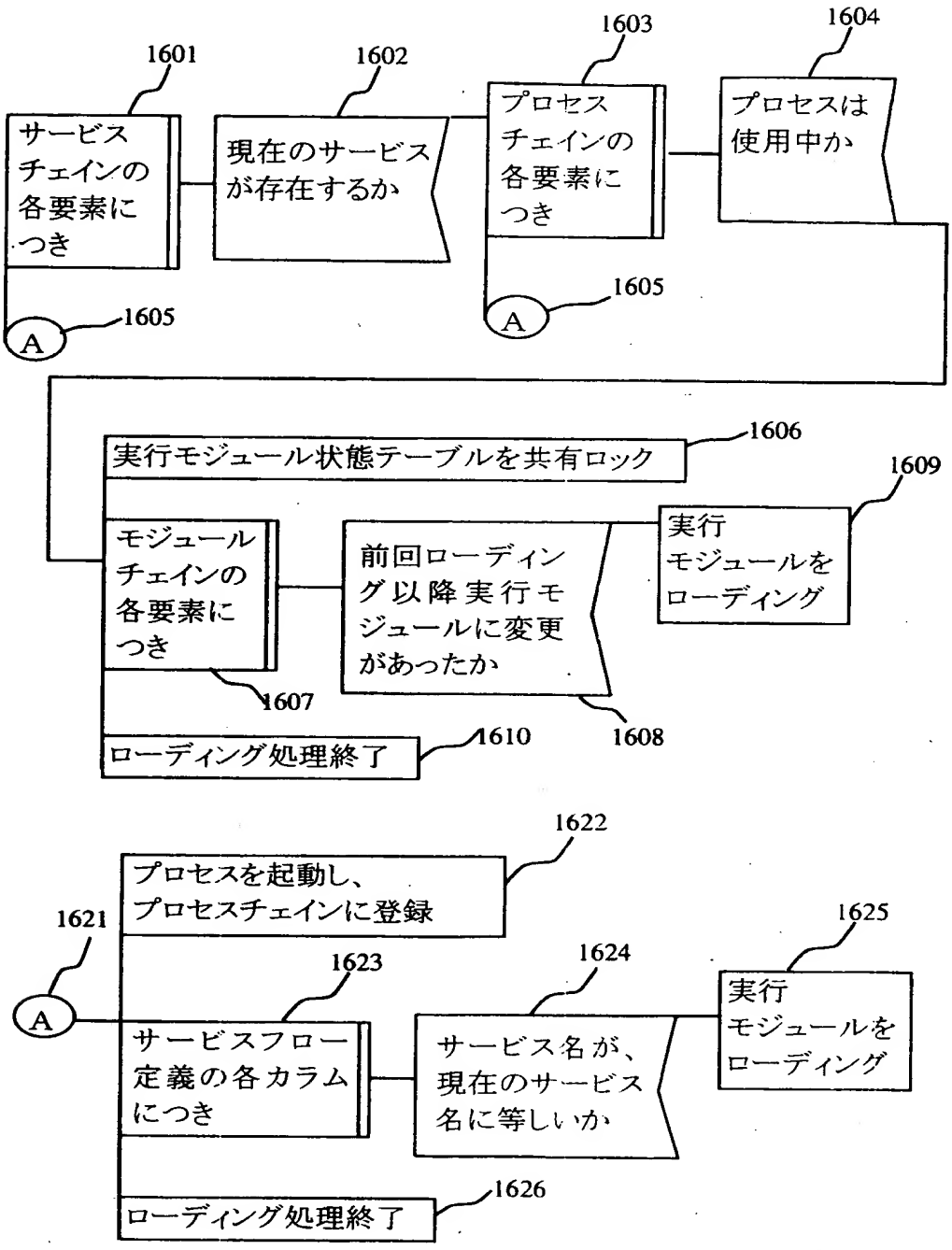
【図１０】


1401
Tul = 初期値
Tbe = 初期値
Pul = NULL
Pbe = NULL
1402
各リクエスト
ヘッダにつき
1403
SLAの
クラスは
U. L. か
1404
Tul > TS
1405
Tul = TS
1406
Pul = (現リクエスト
ヘッダのポインタ)
1407
Tbe > TS
1408
Tbe = TS
1409
Pbe = (現リクエスト
ヘッダのポインタ)
1410
Pul は NULL かつ
Pbe は non-NULL
Pbe は NULL かつ
Pul は non-NULL
Pbe は non-NULL かつ
Pul は non-NULL
1411
Pbe をスケジュール対象とする
1412
Pul をスケジュール対象とする
1413
Tul <
((現在時刻-上限値) + e)
1414
Pul をスケジュール
対象とする
1415
Pbe をスケジュール
対象とする

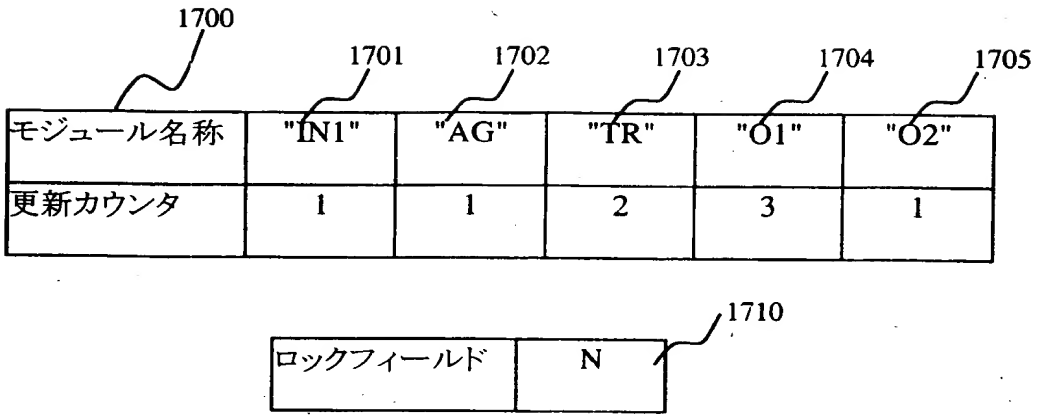
【図１１】


1504
1505
プロセスID
モジュールポインタ
1506
使用中フラグ
1507
後方ポインタ
1508
1509
モジュール名称
後方ポインタ
バージョンカウンタ
1510
1501
1502
1521
サービス名称
プロセスポインタ
後方ポインタ
1503
サービスA
1531
111
Y
1541
1542
1543
"IN1"
"AG"
"O1"
1
1
1
モジュールチェイン
プロセスチェイン
サービスチェイン
1532
112
N
1551
1552
1553
"IN1"
"AG"
"O1"
1
2
1
モジュールチェイン
1522
サービスB

【図１２】

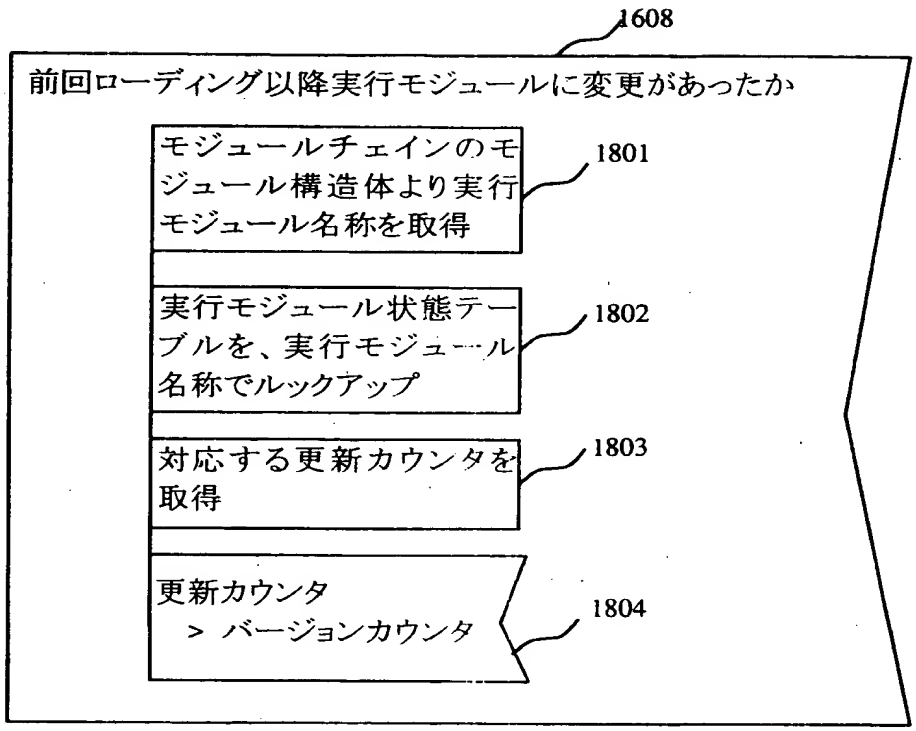

1601
サービスチェインの各要素につき
1602
現在のサービスが存在するか
1603
プロセスチェインの各要素につき
1604
プロセスは使用中か
1605
A
1605
A
1606
実行モジュール状態テーブルを共有ロック
1609
実行モジュールをローディング
モジュールチェインの各要素につき
前回ローディング以降実行モジュールに変更があったか
1607
1608
ローディング処理終了
1610
1622
プロセスを起動し、プロセスチェインに登録
1621
A
1623
サービスフロー定義の各カラムにつき
1624
サービス名が、現在のサービス名に等しいか
1625
実行モジュールをローディング
ローディング処理終了
1626

【図１３】

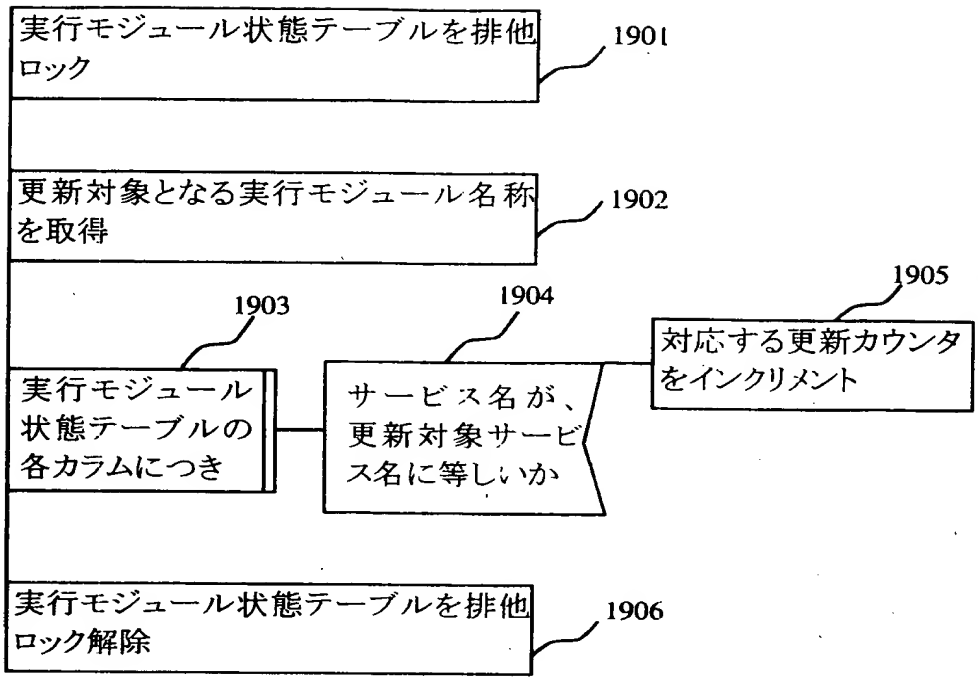


出証特２００１－３０７７４５３

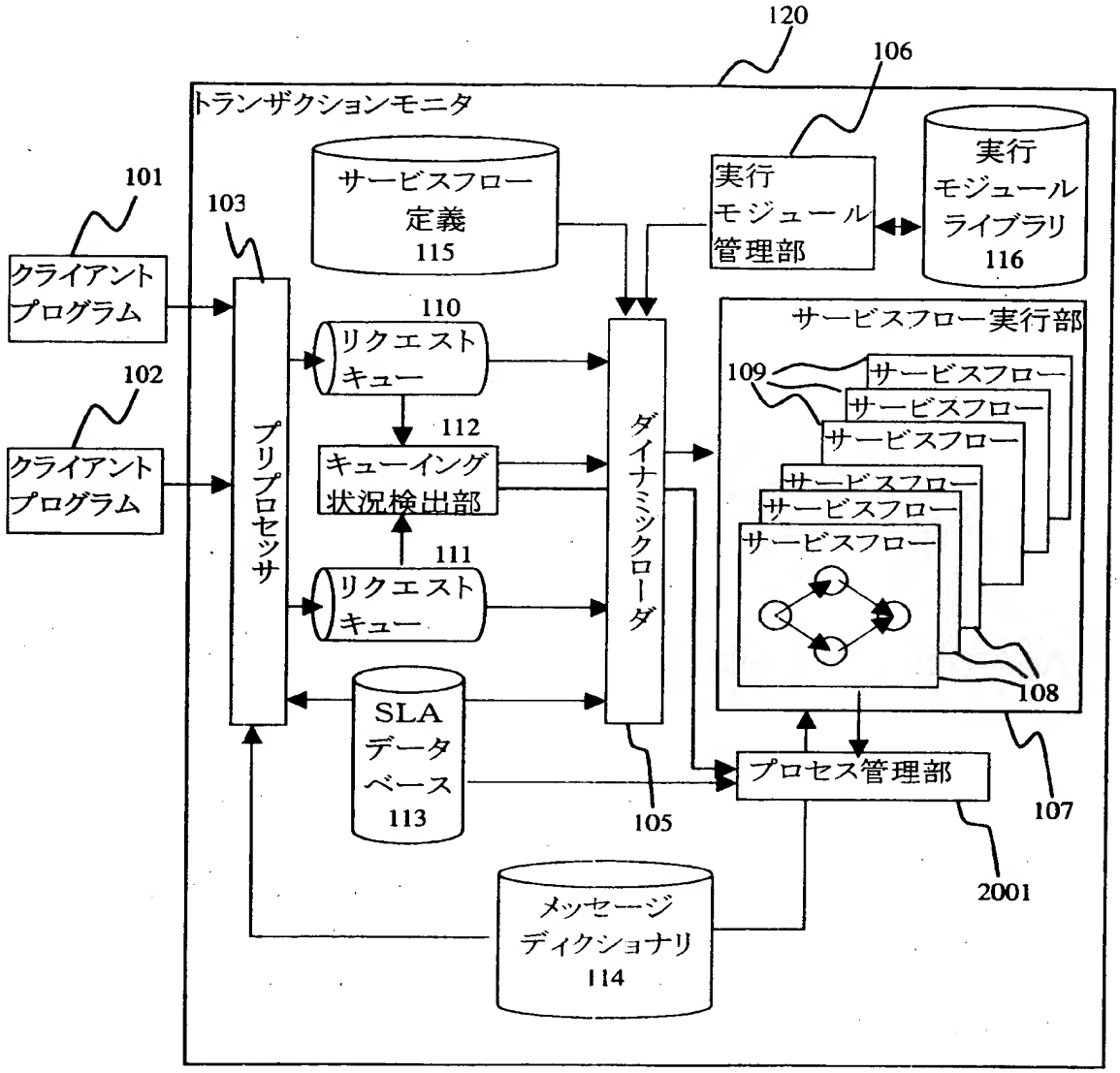
【図１４】


1608
前回ローディング以降実行モジュールに変更があったか
モジュールチェインのモジュール構造体より実行モジュール名称を取得
1801
実行モジュール状態テーブルを、実行モジュール名称でルックアップ
1802
対応する更新カウンタを取得
1803
更新カウンタ
> バージョンカウンタ
1804

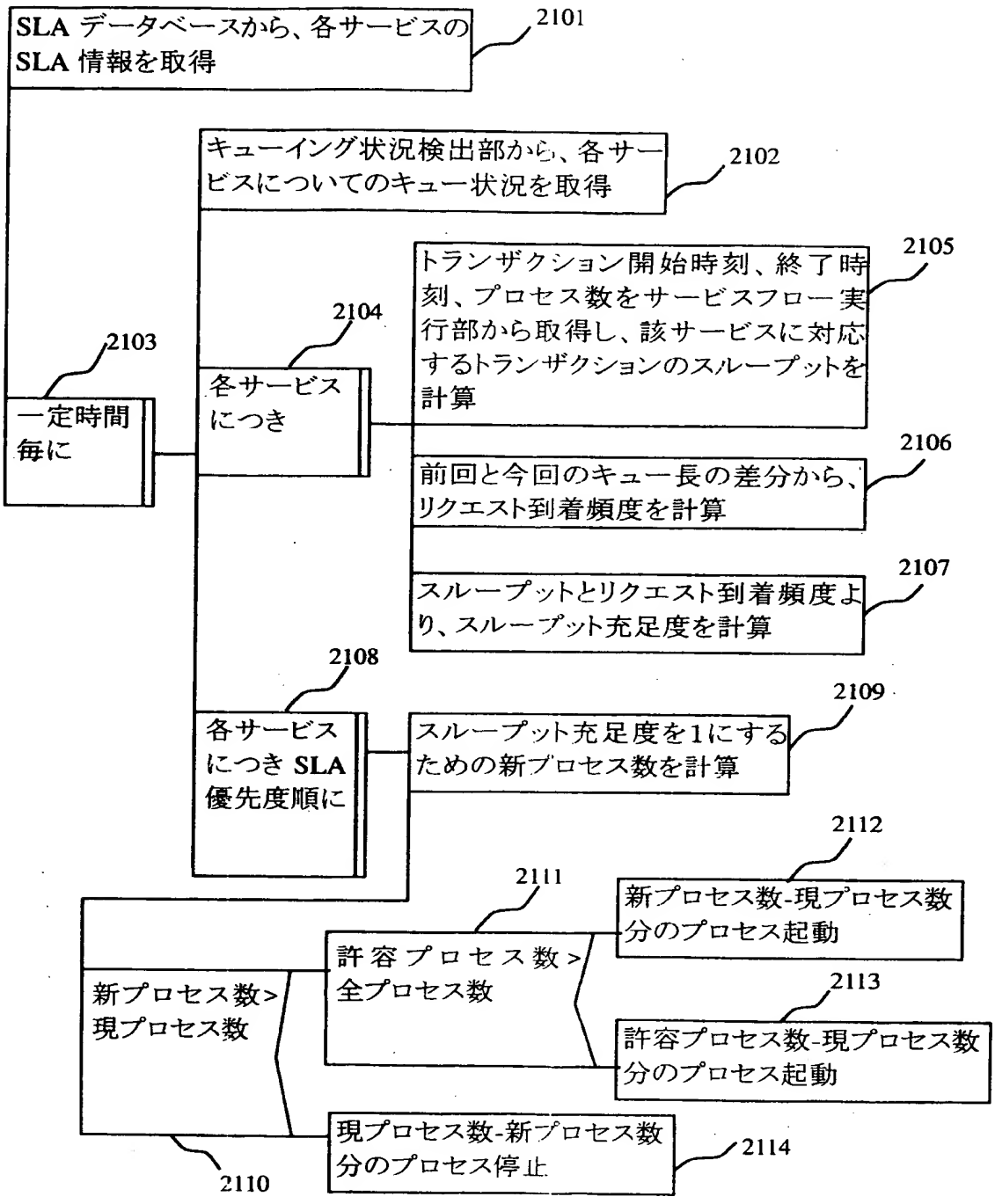
【図１５】


実行モジュール状態テーブルを排他ロック
1901
更新対象となる実行モジュール名称を取得
1902
1903
実行モジュール状態テーブルの各カラムにつき
1904
サービス名が、更新対象サービス名に等しいか
1905
対応する更新カウンタをインクリメント
実行モジュール状態テーブルを排他ロック解除
1906

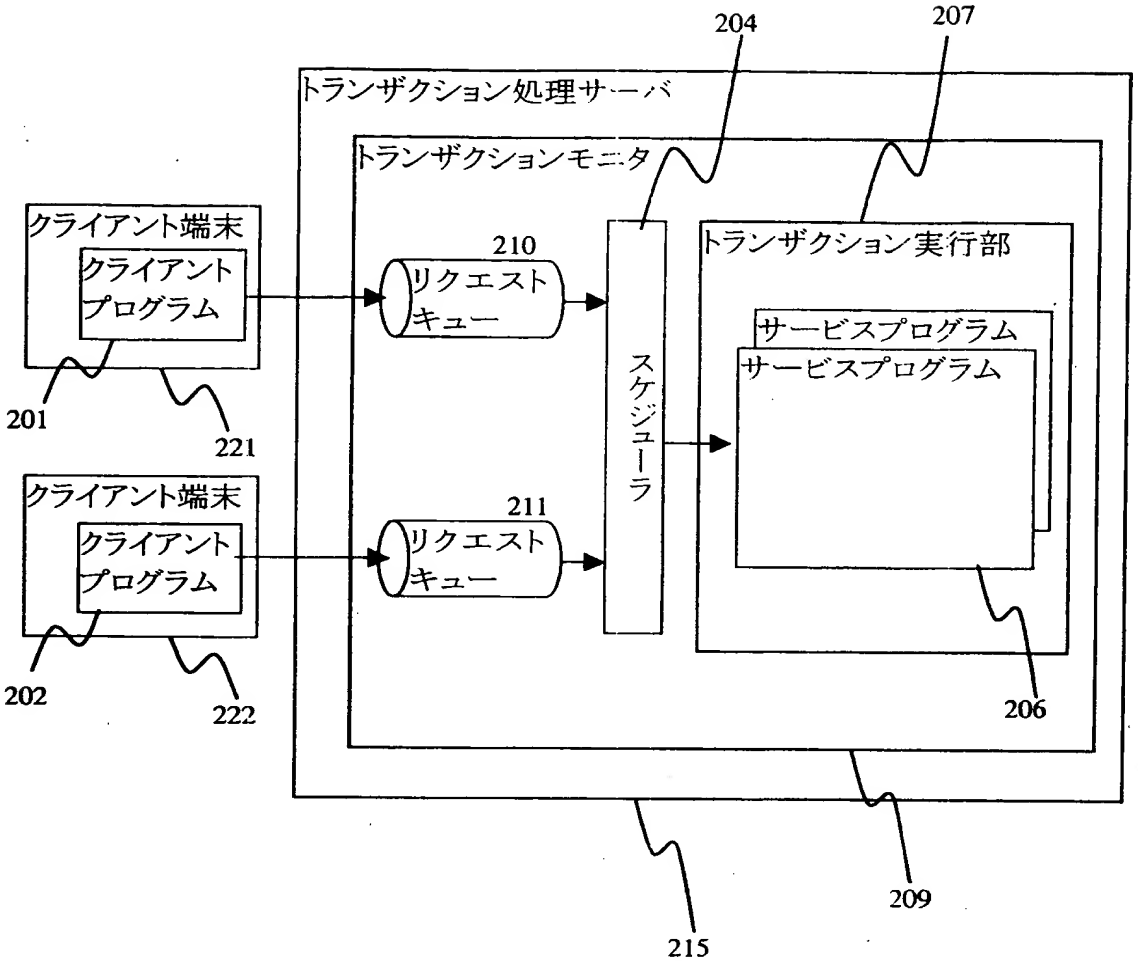
【図１６】


120
106
トランザクションモニタ
101
103
クライアント
プログラム
102
クライアント
プログラム
プリプロセッサ
サービスフロー
定義
115
実行
モジュール
管理部
実行
モジュール
ライブラリ
116
110
リクエスト
キュー
112
キューイング
状況検出部
111
リクエスト
キュー
ダイナミックローダ
サービスフロー実行部
109
サービスフロー
サービスフロー
サービスフロー
サービスフロー
サービスフロー
サービスフロー
108
SLA
データ
ベース
113
105
プロセス管理部
107
2001
メッセージ
ディクショナリ
114

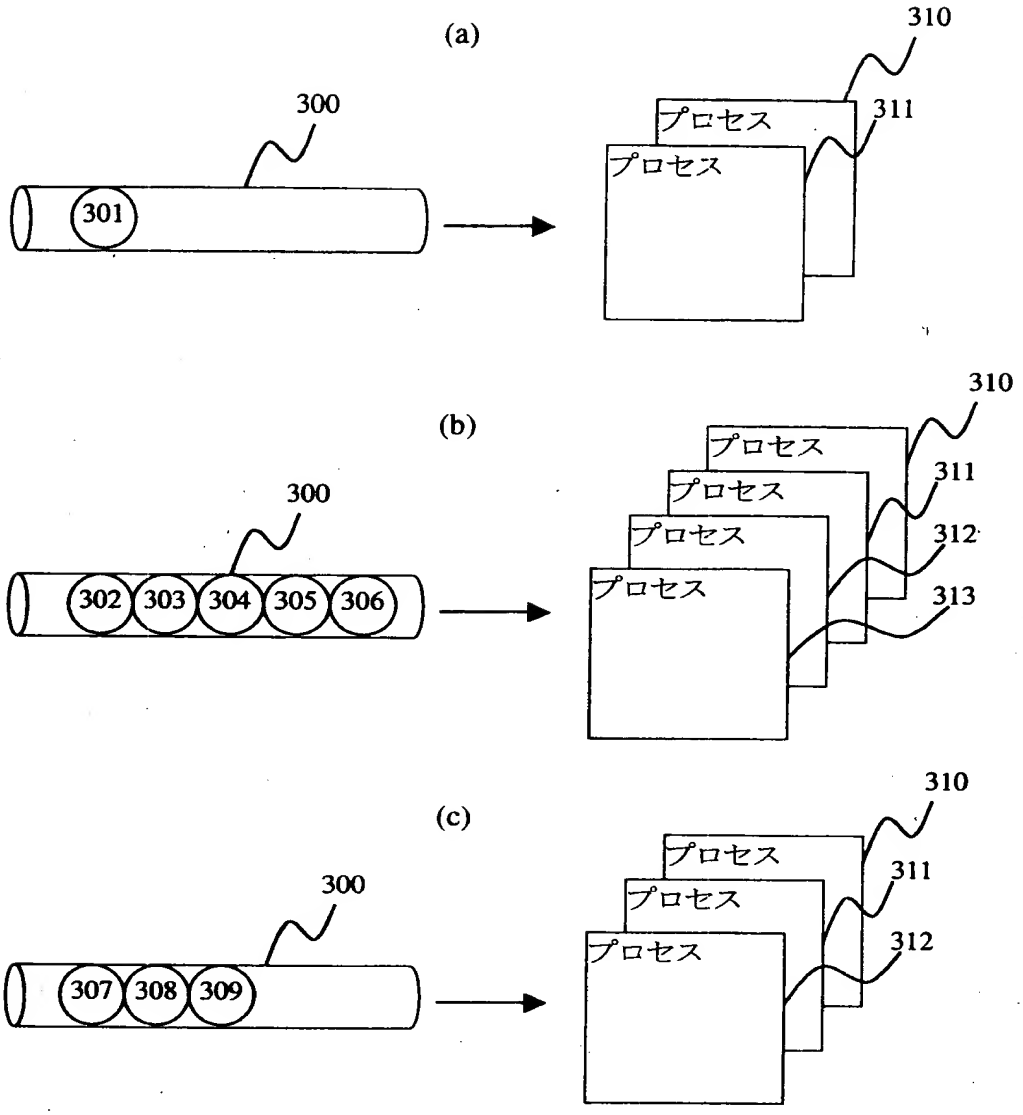
【図１７】


SLA データベースから、各サービスの SLA 情報を取得
2101
キューイング状況検出部から、各サービスについてのキュー状況を取得
2102
トランザクション開始時刻、終了時刻、プロセス数をサービスフロー実行部から取得し、該サービスに対応するトランザクションのスループットを計算
2105
2104
2103
一定時間毎に
各サービスにつき
前回と今回のキュー長の差分から、リクエスト到着頻度を計算
2106
スループットとリクエスト到着頻度より、スループット充足度を計算
2107
2108
各サービスにつき SLA 優先度順に
スループット充足度を1にするための新プロセス数を計算
2109
2112
新プロセス数-現プロセス数分のプロセス起動
2111
許容プロセス数>全プロセス数
2113
許容プロセス数-現プロセス数分のプロセス起動
新プロセス数>現プロセス数
現プロセス数-新プロセス数分のプロセス停止
2114
2110

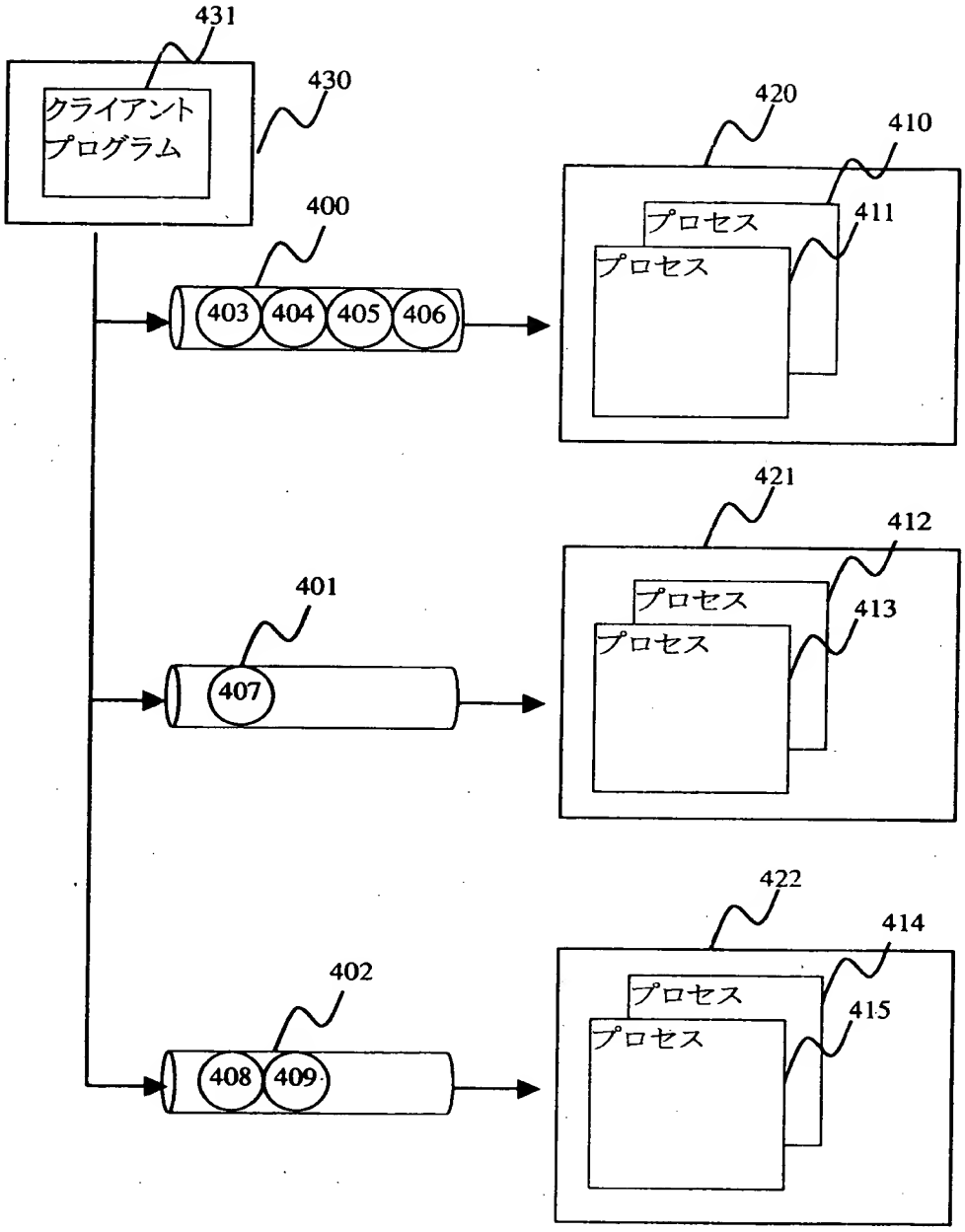
【図１８】


204
207
トランザクション処理サーバ
トランザクションモニタ
クライアント端末
クライアント
プログラム
201
221
210
リクエスト
キュー
スケジューラ
トランザクション実行部
サービスプログラム
サービスプログラム
クライアント端末
クライアント
プログラム
202
222
211
リクエスト
キュー
206
209
215

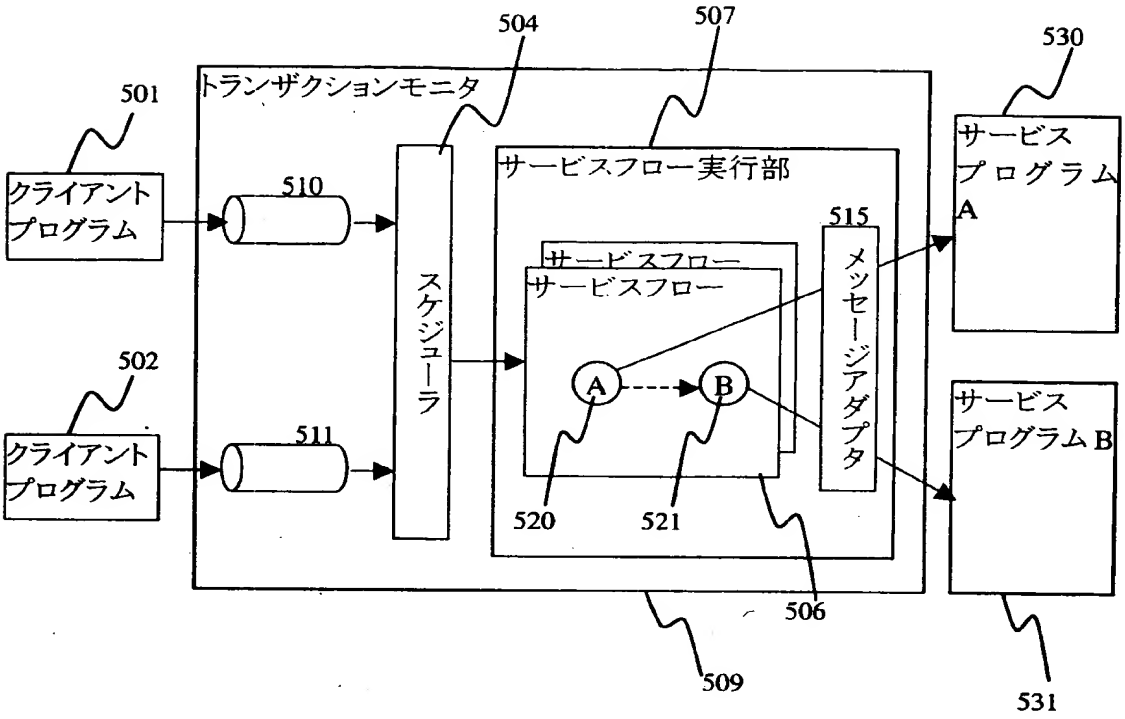
【図１９】


(a)
300
301
310
311
プロセス
プロセス
(b)
300
302
303
304
305
306
310
311
312
313
プロセス
プロセス
プロセス
プロセス
(c)
300
307
308
309
310
311
312
プロセス
プロセス
プロセス

【図２０】


431
クライアント
プログラム
430
420
410
400
プロセス
411
プロセス
403
404
405
406
421
412
401
プロセス
413
プロセス
407
422
414
402
プロセス
415
プロセス
408
409

【図２１】


504
507
530
501
トランザクションモニタ
サービス
プログラム
A
クライアント
プログラム
510
サービスフロー実行部
515
サービスフロー
サービスフロー
メッセージアダプタ
スケジューラ
502
A
B
サービス
プログラムB
クライアント
プログラム
511
520
521
506
509
531

【書類名】　要約書

【要約】

【課題】　トランザクション処理システムにおいて、サービスレベル契約に従って複数のサービスを提供する。

【解決手段】　トランザクション処理システムが提供するサービスに対応して定義された契約情報を格納するＳＬＡデータベース１１３と、クライアントから提供するサービスに対して送付された処理要求を、提供するサービス毎に順序付けて格納するリクエストキュー１１０、１１１と、これらキューイング手段に格納した処理要求の滞留情報を取得するキューイング状況検出部１１２と、クライアントからトランザクション処理システムに投入された処理要求の優先度を、契約情報および処理要求の滞留情報を参照して決定するスケジューラ１０４とを有する。

【選択図】　図１

出証特２００１－３０７７４５３

出　願　人　履　歴　情　報

識別番号　　〔０００００５１０８〕

１．変更年月日　　１９９０年　８月３１日
［変更理由］　新規登録
住　所　東京都千代田区神田駿河台４丁目６番地
氏　名　株式会社日立製作所